
AMMAG
TOUT POUR VOTRE AMSTRAD
HORS SÉRIE
Joyeux Annuaire 1989
AMSTRAD de A à Z
EDITION 1989
H.S. N°9 — 24 F
BELGIQUE : 180 FB — SUISSE : 7,50 FS — CANADA : 7,25 $ CAN
M 1896 - 9 H - 24,00 F-RD
3791896024009 00095

ABONNEZ-VOUS A





La meilleure couverture de l'actualité AMSTRAD



12 N°s + 4 Hors-Série

275 F au lieu de **340 F***

(* Prix de vente au numéro : 12 N°s à 20 F + 4 HS à 25 F)

Vous bénéficiez de 3 avantages

1- ECONOMIE
65 F

2- GARANTIE

Vous pouvez mettre fin à votre abonnement à tout moment. Il vous suffit de nous adresser un courrier avec votre signature. Nous nous engageons à vous rembourser le solde de votre abonnement restant à courir.

3- CADEAU : 2 N°s HORS-SERIE à choisir parmi les N°s parus en 87



1. Amstrad de A à Z



2. Spécial listings



3. Guide des Logiciels



4. Spécial Matériel

Abonnement à AM MAG

☐ Oui, je m'abonne à AM MAG pour 12 N°s + 4 HS au prix de 275 F (au lieu de 340 F)
☐ Europe : 375 F ☐ Airmail : 400 F

☐ Je choisis en cadeau 2 numéros Hors Série : ☐ N° 1 ☐ N° 2 ☐ N° 3 ☐ N° 4
Règlement par : ☐ Chèque Bancaire ☐ Chèque Postal ☐ Mandat

à adresser à :
AM MAG Service Diffusion
5/7, rue de l'Amiral Courbet
94160 SAINT-MANDE

NOM : .... Prénom : ....
Adresse ....
....



décembre 1988

SOMMAIRE

Directeur de la publication : Jean Kaminsky.
Rédaction : Georges Brize, Eric Charton, Bernard Jolivalt, Michel Merlet, Eric Mistelet, Frédéric Nardeau, Guillaume Ponticelli, René Spiegel, François Villon, Henri Wone.
Secrétaire générale de la rédaction : Françoise Kergreis.
Secrétaires de rédaction : Adeline Alphonse, Suan Ajirent.
Maquette : Jean-Jacques Galmiche. **Dépôt légal :** 4e trimestre 1988.
Edité par : Laser Presse S.A., 5-7 rue de l'Amiral Courbet, 94160 Saint-Mandé.
Ce hors-série d'AM-MAG est une publication strictement indépendante et n'a aucun lien avec la société Amstrad.
Imprimé par : RBI et Tima Roto.





A la demande de nombreux lecteurs, nous rééditons ce véritable petit dictionnaire de l'Amstrad. Le débutant (mais aussi l'habitué de la machine) y trouvera une explication claire de nombreux termes qui parsèment son manuel (AMSDOS, Z80, etc.).
Comme les techniques, en informatique plus qu'ailleurs, évoluent rapidement, ce numéro spécial a bien sûr été remis à jour.

AMSDOS

AMSDOS est le système d'exploitation des ordinateurs AMSTRAD de la famille des CPC. Lorsque le CPC 464 fut développé, il fut conçu comme une machine très ouverte, pouvant être complétée par des modèles différents dans une future gamme plus performante. Lorsque une telle optique de conception est formulée, il devient nécessaire de concevoir un système de gestion de l'ordinateur à la fois souple d'utilisation et néanmoins assez rigide pour permettre aux concepteurs d'éventuelles futures machines de modifier des éléments logiciels et matériels sans pour autant supprimer la compatibilité entre les différents modèles.

Dans le cas du CPC, il existe des constantes qui ne varient jamais d'une version à une autre (CPC 464, CPC 6128, CPC 664) :
- BASIC intégré, processeur Z80, mémoire écran de 16 Koctects, avec trois modes, une page de mémoire principale de 64 Koctets, juxtaposée sur une autre page de mémoire variable comprenant AMSDOS en ROM.

En dehors de ces contraintes, la gamme des CPC est ouverte, et il est possible d'étendre la mémoire par blocs de 64 Koctects, de rajouter des ROM exécutables dès la mise sous tension de la machine, un ou deux lecteurs de disquettes en plus de l'unité à cassettes. C'est pourquoi nous avons pu voir la gamme des CPC se modifier, évoluer au gré du marché avec trois modèles différents successifs), tout en restant néanmoins compatible avec les logiciels destinés aux appareils de la gamme inférieure (compatibilité ascendante). Quelques petits problèmes de compatibilité furent néanmoins rencontrés, mais ceux-ci étaient essentiellement dus à un non respect des normes de programmation d'AMSDOS par les concepteurs de logiciels.

Voici les traits principaux d'AMSDOS, et comme vous pouvez le constater, les utilisateurs de CPC sont souvent bien loin d'imaginer la puissance que recèle leur micro. AMSDOS est divisé en plusieurs modules : le système de gestion du hard, le BASIC résident, le DOS pour les lecteurs de disquettes, le système de gestion des cassettes, un système d'instructions améliorées dit de RSX, qui offre aux utilisateurs la possibilité de rajouter des instructions basic sans limitation (celle-ci pouvant être contenue en RAM ou en ROM), un BANK MANAGER qui gère les banques de mémoire additionnelles (sur le CPC 6128). Les programmeurs qui utilisent l'assembleur peuvent exploiter toutes les ressources d'AMSDOS en utilisant des JUMPBLOCKS situés en RAM utilisateur, et qui accèdent à toutes les primitives de bases de la ROM (gestion du graphisme, des lecteurs de disquettes ou de cassettes, gestion des RSX etc., etc.). Ces JUMPBLOCKS permettent à AMSTRAD de faire évoluer sa machine en modifiant la ROM contenant AMSDOS, sans pour cela supprimer la compatibilité entre les différents modèles.

Pour finir, sachez que AMSDOS peut gérer des extensions de ROM (par blocs de 16 K) et qu'ainsi, il devrait permettre de placer de nouveaux langages, ou tous autres types d'applications résidentes (cas de certains logiciels tels ceux commercialisés par la société MAXAM, en Angleterre). AMSDOS fut développé, en même temps que le BASIC, par la société LOCOMOTIVE SOFTWARE.

AMSTRAD

Ce nom est devenu en France, et dans plusieurs pays d'Europe, synonyme d'ordinateurs. Après Apple, Commodore, Sinclair, Oric, cette firme anglaise a repris le flambeau de l'ordinateur domestique (ou "grand public). Présenté sur le marché en même temps que les machines de type "standard MSX", les CPC se sont imposés rapidement ouvrant ainsi la porte aux futurs ordinateurs de la marque : les PCW et dernièrement le nouveau PC.

Amstrad a remporté l'un des plus beau succès dans l'univers de la micro-informatique en vendant plus d'un million de machines (CPC et PCW) en Europe et dans le monde entier. Si l'Angleterre a réussi des scores honorables en ventes de matériels sous la marque Amstrad, précisons que la France et l'Allemagne et plus récemment l'Espagne, sont les pays où la firme s'est particulièrement imposée.

Amstrad, c'est également des chaînes Hi-Fi, des magnétoscopes et des antennes TV qui offrent un rapport qualité-prix très intéressant. D'autres produits basés sur l'électronique de pointe sont à l'étude.
La plupart du matériel que nous connaissons est fabriqué en Corée. Les développements des nouveaux projets se font à Brentwood, au siège de la firme.

Les adaptations pour chaque pays, surtout en matière de micro-informatique, sont réalisés par la société représentant la marque dans le pays concerné (comme par exemple Amstrad en France ou Schneider en Allemagne). Ces sociétés vont également s'occuper de la traduction des manuels, de la promotion des produits et de sa distribution, du service après vente, etc.
Le nom d'Alan Michael Sugar est indissociable de la marque Amstrad. Après avoir vendu des antennes de télévision, il crée Amstrad en 1968 en utilisant ses propres initiales : A.M.S.TRADing corporation. Dur en affaire et très doué, il réussit dans tous les domaines qu'il aborde : le premier ordinateur signé Amstrad, le CPC 464, est une réussite phénoménale. Avec le PCW, le traitement de texte digne de ce nom est mis à la portée de chacun. En 1986, il rachète Sinclair et s'attaque au monde du compatible PC avec le1512 puis il sort le portable PPC. En 1988, la gamme PC 2000, basée sur les micro-processeurs les plus performants du moment (des 80286 et 80386 notamment) l'élève au rang des grands constructeurs.
Méconnu aux Etats-Unis, Amstrad étend son empire en Europe et s'implante solidement en Allemagne, en Espagne mais surtout en France où sa filiale est très active.
Amstrad on en parle beaucoup. Peut-être un peu plus depuis que la publicité passe sur les chaînes de télévision. Des journaux spécialisés aux plus généraux, Amstrad ne passe pas une semaine sans voir son nom cité (en bien ou en mal). Une seule remarque, Amstrad est délibérément absent de toutes manifestations sportives culturelles ou humanitaires à travers le monde. Pourtant c'est aussi, et peut-être, le meilleur moyen d'accrocher encore plus de public. Une affaire à suivre...

A.S.C.I.I.

Inventé par les Américains, ASCII représente les initiales de American Standard Code to Interchange Information. Il s'agit en fait d'un code universel utilisé par beaucoup d'ordinateurs.

La représentation des données alphanumériques c'est-à-dire des caractères tels que l'alphabet, ont avec ce code une représentation aisée.
Chaque caractère est codé en un code de huit bits. Outre l'existence de l'A.S.C.I.I., il y a aussi l'E.B.C.D.I.C. qui est une variante de l'A.S.C.I.I. utilisé par IBM. Il n'est donc pas utilisé par les micro-processeurs sauf si l'on veut s'interfacer avec un terminal IBM. De toute façon on doit pouvoir coder au minimum les vingt six lettres de l'alphabet en majuscule et en minuscule les chiffres de 0 à 9. En général on ajoute le codage au minimum d'une vingtaine de caractères spéciaux. (par exemple le petit bonhomme dont se sert le manuel dans un programme de démonstration). Bien que se code soit universel, sur certains micro-processeurs (6800, 6502...) le nombre de caractères est limité à 128 (de 00 à 127) le huitième bit étant utilisé comme bit de parité. Cette technique est assez utile puisqu'elle permet de vérifier que le contenu d'un mot n'a pas été changé accidentellement. Le fait qu'il n'y ait pas sur le Z80 ce bit utilisé pour la vérification on prend la totalité de l'octet, ce qui donne 256 caractères de disponible (de 00 à 255). En fait, comme sur les autres ordinateurs, on code à partir

de 0 mais les caractères de 0 à 31 sont des caractères dits de contrôle. Ainsi, si on appuie sur la touche ctrl / la lettre G, on obtient le son de la cloche. Quoiqu'il en soit il est toujours utile de retenir que le code A.S.C.I.I. de l'espace est 32. Pour accéder à ces différents codes, il existe deux instructions qui les appellent directement. Tout d'abord CHR $ (X) permet lorsque X prend une valeur de 32 à 255 de représenter le caractère correspondant à X. Par exemple, PRINT CHR $ (66) représentera un B sur l'écran. De même pour PRINT CHR $ (32), l'écran affichera (ou laissera) un blanc. A l'inverse l'autre fonction ASC donne le code décimal correspondant au caractère entre guillemets. Ainsi PRINT ASC ("B") donne 66. De même PRINT ASC (" ") donne 32. Mais comment sont donc définis ces caractères ? Vous savez qu'on peut redéfinir un caractère, par exemple le caractère 240 est une flèche vers le haut. Redéfinissons-le. Chaque caractère est défini sur une matrice de 8 sur 8. Donc 64 points définissent un caractère. Prenons par exem-ple un L comme sur la fig. 1. Il s'agit plus précisément de son codage dans la matrice. Comme c'est indiqué sur la figure les colonnes indiquent le poids des bits. La colonne 0 (en partant de la droite) représente n*2^(0) ; n étant égal à zéro si la case est blanche et un si elle est noirçie. Les chiffres obtenus reflètent la somme totale des cases noirçies en rapport avec leur poids. Pour plus de clarté codons notre L comme exemple :

ligne 1 = 32
ligne 2 = 44 (4+8+32)
ligne 3 = 20 (4+16)
ligne 4 = 40 (32+8)
ligne 5 = 32
ligne 6 = 96 (32+64)
ligne 7 = 168 (2+6+32+128)
ligne 8 = 76 (4+8+64).

Après ce travail on tape : SYMBOL 240,32,44,20, 40, 32, 96, 168, 76. On peut dès maintenant appeler le caractère 240 par PRINT CHR $ (240) et on obtiendra un L délié.

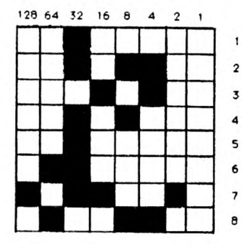


Fig. 1

ASSEMBLEUR

Le langage machine est un langage qui s'adresse directement à la machine à l'aide d'une suite de 0 et de 1. La machine dans le cas des Amstrad, n'est autre que le micro-processeur Z8OA appelé aussi CPU (central processing unit) car il est le cœur de l'ordinateur et dirige tout l'environnement qui lui est destiné (périphérique).

Donc chaque microprocesseur a son propre langage; langage qu'il est plus facile d'apprendre lorsqu'il est représenté par des codes symboliques, rappelant le plus possible l'opération effectuée (1d 1,a = charge ['load' en anglais] le contenu de 1 dans a) qui constituent, tout de suite après le langage machine, le langage assembleur. Au-delà du langage machine interne, on trouve le langage d'assemblage. Celui-ci permet de se dégager de la gestion de la mémoire c'est-à-dire que le programmeur peut nommer les variables qu'il utilise en recourant à des identificateurs de variables qui n'ont plus rien à voir avec leurs adresses d'implantation physique. Par exemple si dans un programme une variable désigne un temps, sous certaines réserves, elle pourra être nommée par l'identificateur de variable temps. En utilisant la machine fictive, au lieu d'écrire 1d a,100000 le programmeur écrira 1da,-temps. Pour que ce soit possible il faut disposer d'un programme de traduction, l'assembleur, capable de transformer un programme écrit en langage d'assemblage, en un programme équivalent au précédent en langage machine interne, exécutable par l'ordinateur.

Ainsi le programme conçu et écrit par le programmeur est le programme source. Ce programme est considéré comme une donnée lors de l'exécution de l'assembleur, qui le traduit et génère un programme objet en langage machine équivalent du point de vue algorythmique au programme source. Ce programme objet peut alors être exécuté pour obtenir les résultats cherchés à partir des données de départ (voir fig. 1). Après avoir introduit en mémoire centrale

l'assembleur et le programme source on peut lancer l'exécution de l'assembleur qui fabrique le programme objet. On a donc en mémoire centrale durant la phase d'assemblage les trois programmes (voir fig. 2).
En théorie la phase d'assemblage est fondée sur les deux types d'assembleurs suivants :
- les uns font tout le travail qui leur est assigné en une seule fois, en un seul passage; ce sont les assembleurs à une passe.
- il y a ceux qui font le travail en deux fois, ce sont les assembleurs à deux passes.
De façon schématique, l'assembleur doit commencer par analyser chaque instruction du programme source. Il vérifie que le code opération correspond bien à un code connu qu'il peut retrouver dans une table des codes opérations dont il dispose et qui lui permet de connaître la valeur binaire correspondant à ce code opération. Il vérifie que l'identificateur de variable figurant dans l'instruction respecte bien les règles définies pour le langage d'assemblage comme par exemple le nombre de caractères utilisés dans l'identificateur, le type de ces caractères. Il se construit une table des identificateurs variables.
Soit cet identificateur existe déjà et il suffit de lui affecter l'adresse retenue pour la donnée, soit il n'existe pas encore et l'assembleur affecte l'adresse physique d'un mot à cet identificateur. Il peut alors fabriquer les instructions en langage machine. Cela revient à dire que le rôle de l'assembleur est tout d'abord de vérifier si les instructions du programme source sont correctes du point de vue syntaxique puis de fabriquer les instructions du programme objet. Une fois définies, les adresses de la mémoire centrale correspondent aux variantes utilisées dans le programme source. En fait les

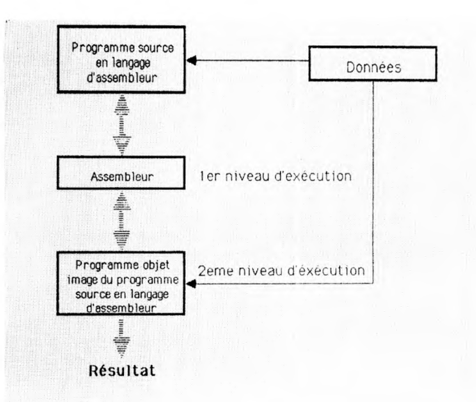


Fig 1

possibilités d'un langage d'assemblage sont plus importantes mais leur présentation dépasserait le cadre de cet article. Cependant le langage d'assemblage est encore utilisé pour réaliser des logiciels chez les constructeurs d'ordinateurs, pour concevoir des systèmes automatiques à base de microprocesseurs, pour fabriquer des sytèmes de contrôle de processus, etc.
Dans les applications de gestion sur une petite ou sur une grosse machine, il n'est presque plus utilisé et sert seulement pour écrire des programmes devant être optimisés quant à leur temps d'exécution, comme par exemple dans certaines opérations de télétraitement.
Voici un exemple plus parlant de l'utilité de l'assembleur : l'instruction *return* en *basic* est exécutée en 0,6 millisecondes alors que l'instruction correspondante en langage assembleur ne dure que 2.5 microsecondes. La programmation du graphisme haute résolution est d'autre part trop lente en *basic*, de sorte que l'assembleur s'impose pour les jeux ou les graphiques de gestion par exemple.

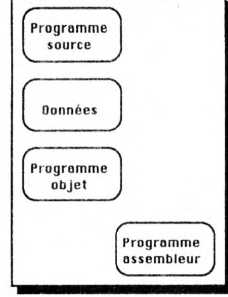
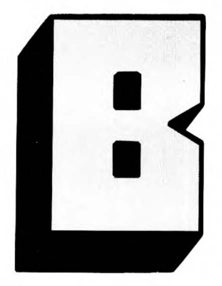


Fig 2

Normalement un utilisateur n'aura jamais à manipuler ce type de langage. Ecrire un programme en assembleur exige que le programmeur connaisse bien la structure de la machine sur laquelle il travaille, cela reste long et difficile car étant le langage le plus proche de la machine, cela présente pour le programmeur l'inconvénient de devoir penser de façon très abstraite pour comprendre ce langage.

PONTICELLI Guillaume



BASIC

B comme Basic. Conçu en 1965 au "Darmouth college" d'HANOVER par un groupe d'enseignants et d'étudiants, sa signifiation est la suivante : "Beginners All-purpose Symbolic Instruction Code".

Le mot symbolique est important car il s'oppose au langage absolu. Puisque l'ordinateur traite les informations sous forme binaire, le langage le plus proche de la machine est justement le langage absolu. Ce langage possède un code, par exemple alphanumérique (codification hexadécimale), qui est arbitraire et souvent peu facile à retenir. On a donc créé un autre langage appelé langage symbolique, en remplaçant systématiquement les codes absolus par des groupes de chiffres ou de lettres rappelant si possible la nature de l'opération. Nous voici donc au niveau "assembleur".

Puis pour faciliter l'utilisation de la machine sont nés des langages dits évolués qui possèdent leur propre syntaxe et leur propre vocabulaire. Parmi un grand nombre de ces langages évolués, on trouve le *basic*.

C'est un langage qui contrairement à notre langage parlé ne possède que très peu de mots, environ deux cents selon les différents basic, pour pouvoir résoudre un problème quelconque. Très proche de notre syntaxe naturelle, il permet à n'importe quel novice de concevoir des programmes simples voire très compliqués, ceci étant bien entendu relatif aux applications qu'on demande à l'ordinateur. il permet en tout cas d'élaborer des programmes appelés "à structure séquentielle non cyclique ou cyclique". Les structures cycliques introduisent une instruction appelée boucle ; par exemple les boucles FOR ... NEXT, WHILE ...WEND. L'exemple de GOSUB ... RETURN (go to subroutine) n'est pas cité comme boucle car on peut le mettre dans les principes dits de récursivité même si certaines mauvaises langues affirment que le BASIC n'est pas un langage récursif.

En gros on peut dire que le Basic a les propriétés suivantes : lent en exécution (soyons cependant relatif dans nos jugements), moyen en gestion, bon en calcul scientifique. Travaillant avec un interpréteur, il est aisé à apprendre car quelques heures suffisent à bien le manipuler. Utilisé au départ sur des systèmes en temps partagé, il a commencé à être introduit à la fin des années soixante dans de grandes entreprises pour que les utilisateurs puissent se composer eux-mêmes des programmes. Il a connu un véritable succès lorsque les petits ordinateurs se sont développés de façon considérable.

En effet pour les constructeurs, il est simple à implanter et pour les utilisateurs, il donne l'illusion de pouvoir écrire des programmes très rapidement. Il est sans conteste le langage le plus utilisé sur les petits systèmes informatiques. Sur les mêmes matériels on trouve aujourd'hui à la fois des interpréteurs pour la mise au point des programmes et des compilateurs qui permettent de garder l'image exécutable en langage machine.

UN LANGAGE INTERACTIF ET INTERPRETE

Conçu à l'origine comme un sous-noyau de FORTRAN il en a les défauts : il ne conduit pas le programmeur à avoir une démarche algorythmique correcte. De plus il n'a été normalisé qu'en 1979 ce qui a impliqué une très grande variété de BASICS sur le marché. Ce qui fait que les programmes sont intransposables mais de plus lors d'un changement de machine il faut en apprendre les nouvelles particularités. Cependant tous les BASICS ont les caractéristiques suivantes : BASIC est un langage de programmation de haut niveau.

C'est un langage interactif et interprété, c'est-à-dire que chaque instruction est traduite en langage machine et exécutée immédiatement lorsqu'elle est reconnue comme étant correcte. S'il y a des erreurs (de syntaxe par exemple, un oubli de point virgule !), elles sont ainsi détectées tout de suite et l'on peut les corriger. Son caractère interactif facilite l'accès à l'ordinateur sans qu'il soit nécessaire de se familiariser avec les idiosyncrasies de systèmes d'exploitation des ordinateurs : cartes contrôles aux langages abscons pour les débutants, messages d'erreurs bien souvent peu explicites, délais d'attente. Il faut noter cependant que le langage peut être utilisé également par des professionnels qui veulent tester rapidement un

algorythme, traiter un problème de mathématiques ou de statistiques élémentaire ou même consulter une petite base de données personnelle.
Il existe cependant de nombreux détracteurs du langage (c'est qu'il intéresse quand même sinon on n'en parlerait pas) : il est ancien et ne permet pas de structurer les programmes, son caractère interactif peut donner des habitudes de programmation détestables. D'autre part les programmes écrits en BASIC sont exécutés lentement et c'est un langage à base anglaise ce qui les rend moins adaptés à l'enseignement dans les pays non anglo-saxons. Pour ce qui est des premières critiques, elles sont dans une large mesure justifiées : le caractère interactif peut donner des programmes mal construits mais que l'on peut considérer comme des brouillons que l'on jette et que l'on peut éventuellement utiliser comme des bases pour des programmes de manière structurée dans un autre langage. En ce qui concerne les dernières critiques, elles ne sont que partiellement valides. La vitesse d'exécution n'a plus de sens lorsque l'utilisateur obtient une réponse quasi instantanée à son niveau. En effet dans bien des cas la réponse est obtenue de façon plus rapide par l'utilisateur d'un micro-ordinateur que s'il avait dû passer par un système de temps partagé où il n'est pas le seul utilisateur. Le caractère anglophone du langage est secondaire dans la mesure où l'on peut écrire en basic francophone moyennant un petit effort de traduction. Ceci dit BASIC reste un langage pratique simple et disponible sur tous les micro-ordinateurs du marché.

PONTICELLI Guillaume

BINAIRE

Les principaux circuits composant un ordinateur sont des circuits logiques (tel le micro-processeur, le gate array, les décodeurs...) qui fonctionnent en tout ou rien, c'est-à-dire 1 ou 0. Il est donc important en informatique de bien savoir manier le binaire, c'est ce que nous allons voir dans cet article.

A l'époque des premiers ordinateurs, les langages évolués, tels le BASIC ou le PASCAL, n'existaient pas, on programmait alors en langage machine, c'est-à-dire avec des 0 et des 1. Le binaire était donc, à l'époque, le seul langage de programmation. De nos jours, il n'est plus nécessaire de connaître le binaire pour pouvoir programmer. Toutefois, il est utile lors, par exemple, de l'utilisation de l'instruction SYMBOL qui permet de redéfinir un caractère. Vous savez que les ordinateurs AMSTRAD CPC et PCW fonctionnent avec le microprocesseur Z80 qui est un 8 bits (voir l'article s'y rapportant). Un bit est la plus petite unité mémoire qui ne peut prendre que 2 valeurs : 0 ou 1. Un groupe de 8 bits s'appelle un octet et nous allons voir ensemble comment on code une valeur en base dix sur un octet. Les bits d'un octet sont numérotés de droite à gauche de B0 à B7. On appelle BO le LSB = Low Significant Bit (bit le moins significatif), et B7 le MSB = Most Significant Bit (bit le plus significatif). Chaque bit a un "poids", c'est-à-dire une valeur qui lui est propre. Le bit Bi, lorsqu'il est à 1, vaut 2^i (^ signifie puissance) ; donc, si B6=1, il représente à lui seul 64 en décimal. Lorsque plusieurs bits d'un octet sont à 1, on ajoute leurs valeurs respectives. Donc, si on a l'octet 00010101, on a B0, B2, B4 à 1, donc 2^0+2^2+2^4, ce qui fait 1+4+16 = 21 (n'oubliez pas que x^0 = 1 quel que soit x non infini). Le nombre de combinaisons possibles sur un nombre n de bits est égal à 2^n. Ainsi, dans le cas d'un octet, le nombre maximum de combinaisons est égal à 2^8, soit 256. On peut donc coder sur un octet un nombre compris entre 0 et 255 (le 0 est bien évidemment représenté lorsque tous les bits sont à 0). Essayez de coder 170 en binaire. Si vous trouvez 10101010, c'est que vous avez compris.
Nous venons de voir comment coder un nombre entier, positif, sur un octet. Nous allons maintenant aborder la représentation d'un entier qui peut être négatif ou positif, c'est ce que l'on appelle le binaire signé. Ici, le MSB représente le signe du nombre : 0 = nombre positif, 1 = nombre négatif. Dans octet, c'est donc le bit B7 qui détermine le signe. Puisque ce bit est réservé, on n'a plus que 7 bits pour représenter notre nombre qui peut ainsi être compris entre +127 et -128. Le codage d'un nombre positif sur un octet s'effectue comme précédemment avec le bit B7 à 0. Le codage d'un nombre négatif est plus complexe. IL faut d'abord coder le nombre sur 7 bits comme s'il était positif avec B7 = 0. Puis, il faut complémenter le tout, c'est-à-dire qu'un 0 deviendra un 1 et

vice-versa. Il ne reste plus qu'à ajouter 1 à ce nombre. Le voici codé sur un octet. Ce procédé appelé "le complément à deux" est très utilisé par tous ceux qui programment en langage machine lors de calculs de sauts relatifs comme dans l'instruction JR sur le Z80. Supposons que nous voulons coder -1. 1 en binaire s'écrit 000000001. On complémente : on obtient 11111110. On ajoute 1 : cela donne 11111111. C'est ainsi que - 1 est codé sur un octet. Essayez de coder - 27, si vous obtenez 11100101, c'est que vous avez compris, bravo, vous êtes doués.
Pour le codage d'un nombre décimal (c'est-à-dire un nombre à virgule), reportez-vous à l'article correspondant, et soyez à l'aise dans le binaire, ce ne sera pas de trop !

BIOS

Décidément, le jargon informatique fourmille de termes barbares ! Faut-il en déduire que les informaticiens sont des maniaques du langage, ou qu'ils cherchent par tous les moyens à rendre leur discipline inaccessible ? Et bien non, en fait BIOS ne fait pas allusion à un ordinateur bionique, ou à un quelconque type de cerveau artificiel, mais signifie "Basic Input Output System".

Ciel encore de l'Anglais ! Pas de panique, en voici la traduction : Bios traduit veut dire "système d'entrées sorties de base". On retrouve en fait des Bios dans tous les systèmes d'exploitation, qu'ils soient CP/M 80, CP/M 3.0, MS-DOS, ou CP/M 86. En effet, quel est le principe de base d'un système d'exploitation ? Il s'agit d'un logiciel qui, à l'origine devait être portable, c'est-à-dire capable de migrer d'une machine à une autre, avec pour seule limitation dans l'architecture de ces machines qu'elles aient le même processeur, par exemple 8080 pour CP/M 80, Z80 pour le CP/M 3.0 (en offrant par la même occasion une compatibilité ascendante avec le CP/M 80), 8086 pour CP/M 86, 8088 ou 8086 pour MS-DOS (par la suite MS-DOS put être implanté sur des 80186, 80286 et 80286).

On a un peu tendance à être induit en erreur lorsque l'on parle de systèmes d'exploitation, à cause du MS-DOS de l'IBM PC. Dans le cas des PC, le standard est dû à une compatibilité HARD et SOFT, mais il existe également des machines dites compatibles MS-DOS. Tout ceci peut vous paraître un peu confus mais va s'éclaircir lorsque vous aurez lu les quelques lignes qui suivent.
Prenons l'exemple du CP/M 80 : il existe sur des machines radicalement différentes (cas des CPC par rapport au PCW), mais qui sont néanmoins compatibles entre elles sous ce système, ceci parce que les programmes font des appels aux vecteurs du BIOS, sans utiliser directement les caractéristiques propres à l'architecture des machines. Par exemple, pour afficher un caractère à l'écran, nous n'utiliserons pas, sous CP/M, la mémoire écran mais le vecteur destiné à l'affichage, ainsi, un même programme pourra s'exécuter sur un PCW et un CPC, qui ont pourtant des processeurs vidéo très différents. Pourquoi cela ? Parce que le BIOS aura dans les deux cas des points d'accès en mémoire strictement identiques, ainsi que des conditions d'entrées parfaitement compatibles (Chargement des mêmes registres, ou adresses de variables systèmes situées aux mêmes endroits). Un fois que la main est donnée à ce vecteur par votre programme, les routines du BIOS, quelles que soient les machines, gèrent les caractéristiques propres du système pour effectuer une action, et peuvent donc être radicalement différentes, en restant toutefois parfaitement compatibles. Bien sûr, un tel procédé impose des désagréments :
- limitation dans l'évolution des machines (ce phénomène se retrouve de toute façon dans n'importe quelle tentative de standardisation) ;
- limitation des performances ;
- gestion d'écran peu performante.
Sur les PC 1512 et compatibles, la notion de BIOS par la force des choses a complètement été transformée. On désigne par BIOS, dans le cas des PC, non pas le système d'entrées-sorties de base du MS-DOS, mais celui des entrées-sorties contenues dans la ROM, accessible par l'intermédiaire de vecteurs d'interruption. Ainsi, on a assisté à une curieuse mutation de la notion de BIOS, MS-DOS devenant un système d'exploitation figé (n'ayant donc pas à prendre en compte les caractéristiques de la machine par exemple le CP/M du CPC 6128 ne fonctionne pas sur un PCW 8256, mais le MS-DOS de l'AMSTRAD PC tourne sur n'importe quel

compatible X ou Y, voire même sur un PC IBM), la ROM seule variant d'une machine à l'autre pour tirer partie de toutes les spécificités de celle-ci (le ROMBios de l'AMSTRAD PC ne fonctionne pas sur un IBM PC).

BIT

Le bit est la plus petite entité numérique que l'on trouve dans un micro-ordinateur. Effectuons pour décrire ce qu'est un bit, un bref récapitulatif sur le calcul en informatique.

En informatique, un mot est composé de deux octets. Un octet est égal à 8 bits. Pour connaître simplement et rapidement le nombre de combinaisons possibles d'un élément en informatique (le nombre de valeurs qu'il peut prendre) il suffit de suivre la règle suivante : 1 bit est un élément simple. Il peut prendre deux valeurs : 1 ou 0. Si un octet est composé de 8 bits, nous avons donc accès à 2 puissance 8 combinaison (notation 2**8), soit 256 combinaisons possibles, le 0 existant en informatique, un octet peut prendre une valeur situé entre 0 et 255. le système de calcul avec les bits est donc très simple. Si l'on vous parle d'un registre sur 2 bits, vous effectuez le calcul de 2**2 et vous obtenez 4 combinaisons. Imprégnez-vous bien de ces calculs, car en informatique vous les retrouverez présents partout, à tous niveaux, et dès que vous commencez à faire de la programmation dite système, c'est toutes les deux lignes !!
Prenons l'exemple d'une mémoire écran. La résolution couleur d'un ordinateur dépend directement de celle-ci, et est encore une fois liée au bit. Si vous avez un seul bit par pixel, celui-ci ne pourra être affiché qu'en deux couleurs : allumé ou éteint. Pour deux pixels, 4 couleurs seront disponibles et ainsi de suite. A titre indicatif, le MODE 2 des CPC prend 1 bit par point (mode 640/200 en 2 couleurs, et le MODE 0, 4 bits par point (mode 160/200 en 16 couleurs). Idem pour les CPC, le mode maximum donne une résolution de 640/200 en 16 couleurs, soit 4 bits par point, soit un espace de mémoire vidéo, par exemple, de 256 K octets dans ce mode. Dernière utilisation du bit, dans le cas de saut dans le bios d'une machine, bien souvent à des fins d'économie de mémoire, un registre de 16 bits contient plusieurs types d'informations différentes. Il faudra donc pouvoir travailler au niveau du bit sur ce registre, ce que permettent des processeurs comme le Z80 ou le 8086, mais ce que ne peut pas faire le 8080 d'INTEL par exemple.

BUG

Ce terme, d'origine anglo-saxonne, signifie, dans le jargon informatique, erreurs de programmation. Vous avez certainement remarqué qu'un programme présentait souvent un défaut que l'on doit certainement à un bug du logiciel en question.

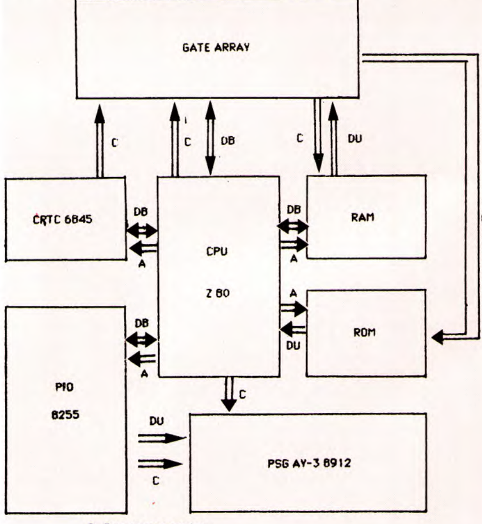
connaissance du langage que l'on utilise, ainsi qu'une bonne méthode de recherche d'où une bonne analyse de la structure du programme.
Même les meilleurs logiciels professionnels contiennent des bugs qui n'apparaissent qu'au bout de nombreuses heures d'utilisation. Rassurez-vous, ce n'est pas toujours le cas et, de plus, ces logiciels sont tout de même très performants. Il suffit, lors de l'utilisation de ces logiciels, que l'on travaille sur un point particulier pour trouver la faille qui n'est, malheureusement pour les programmeurs, lorsqu'ils testent leur logiciel, pas simple à détecter. Cela dit, il ne faut pas mettre une erreur de manipulation sur le dos d'un bug !
Lorsque l'on tape un programme, à moins qu'il ne s'agisse d'une petite routine de deux ou trois lignes, celui-ci ne fonctionne que très rarement du premier coup. Ne serait-ce que par les fautes de frappe. Celles-ci peuvent être faciles à déceller si elles provoquent l'affichage d'un message d'erreur de syntaxe comme le SYNTAX ERROR IN LINE XXXX en BASIC. Ici, la correction ne pose pas trop de problème, à condition que l'on connaisse la syntaxe des commandes du langage que l'on utilise.

En fait, les bugs les plus difficiles à corriger, qui sont d'ailleurs redoutés par les programmeurs, sont ceux qui ne provoquent pas de messages d'erreurs (dans le cas des langages évolués bien sûr, car en langage machine, il n'y a pas de messages d'erreurs). Il peut s'agir de l'utilisation d'une mauvaise variable, ce qui peut être causé par une faute de frappe lors de la programmation, ou par des indices de boucles mal calculés.

BUS

La notion de bus est très importante dans un ordinateur puisqu'elle permet (entre autres) sans ne rien connaître sur la machine de savoir s'il s'agit d'un microprocesseur huit, seize ou trente deux bits.

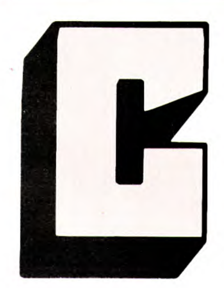
Mais revenons au quotidien (juste un moment !!) Afin de mieux comprendre il faut avoir présent à l'esprit que le bus ou autobus est un engin qui, sur la demande d'un passager s'arrête pour laisser descendre l'intéressé. L'ordinateur travaille de la même façon. On peut voir dans tous les ordinateurs du marché trois sortes de bus : le bus de données (data en anglais) est constitué de huit fils connectés aux principaux circuits concernés. Ainsi, le contrôleur vidéo par exemple possède ses huits fils de données (pattes 26 à 33) ; le gate array également (patte 24 à 31), les ram, la rom, le pio, le générateur de son et le Z80. C'est à ce niveau que l'on voit que le Z80 est un microprocesseur huit bits, c'est-à-dire qu'il peut gérer ou envoyer huit informations simultanément. Pensez aux microprocesseurs trente deux bits ! Le bus de données à la particularité d'être bidirectionnel (deux sens de circulation) c'est-à-dire que deux ou même trois circuits peuvent mutuellement, pas en même temps, s'envoyer des données ; pour le CPC464 on peut récupérer les données du bus sur les connecteurs arrières ; sur la sortie imprimante D0 à D7 seront respectivement du pin 2 au pin 9. Mais les mêmes données sont aussi sur le grand connecteur du pin 26 au pin 20. Par exemple, si le microprocesseur envoie à l'imprimante le caractère A.S.C.I.I. de l'espace (32 en décimal et 00100000 en binaire) cette donnée ira sur le bus de données connecté à tous les autres. Mais cette donnée valable uniquement pour l'imprimante ne sera pas lue que par elle ; pourquoi cela ? C'est là que nous découvrons l'existence d'un second bus appelé bus d'adresses. Pour notre cas, c'est-à-dire 64k octets on devra adresser 64*1024 = 65536 octets ou lignes contenant chacune huit mots de un caractère ou un mot de huit caractères.Pour adresser toutes ces lignes, (on compte à partir de 0, ce qui nous fait



C= Bus de commandes
A= Bus d'adresses
DU= Bus de données, celles-ci ne circulent que dans un sens
DB= Bus de données, celles-ci circulent dans les deux sens

Shéma représentant l'organisation interne des CPC

aller jusqu'à 65535), il est donc nécessaire d'utiliser 2^(n) : 65535 avec n = nombre de fils. On calcule LOG 2^(n) = LOG (65535)
n*LOG (2) = LOG (65535)
n = LOG (65535)/ LOG (2) = 16. Il faut donc 16 fils pour adresser 65536 lignes. Il va sans dire que c'est le microprocesseur Z80 qui commande ces adresses. on comprend alors que la particularité du bus d'adresse c'est d'être unidirectionnel (sens unique). C'est par le bus d'adresse que le Z80 affirme sa suprématie. Prenons l'exemple des rams (l'imprimante n'est pas adressée mais reçoit un signal à l'état bas lui permettant de lire les données). Si l'on veut mettre 255 à l'adresse &49A1, on fera POKE &49A1,255. Cette instruction a l'avantage d'être très vite interprétée. Sur le bus de données on aura 1111111111 et sur le bus d'adresse 0100100110100001. Tant que la ligne de l'adresse &49A1 n'est pas adressée, les données restent sur le bus sans être lues. Pour adresser correctement et sans risquer de valider un autre circuit, il est nécessaire de bien connaître les caractéristiques des circuits que l'on utilise. Ainsi le PIO est validé par les bornes NON CS, A1 et A0 respectivement au pattes 6,8, et 9. Les constructeurs ont décidé d'y connecter les fils d'adresses A11, A9 et A8. Pour sélectionner ce circuit il faut NON CS = 0, A1 = 1, A8 = 1. L'adresse la plus haute étant (en considérant les autres fils à 0) &0B00. On peut récupérer les adresses sur le grand connecteur des pins 2 à 18 (A15 a A0 respectivement). On pourra d'ailleurs s'amuser à l'aide d'un circuit RC de bonne constante de temps et d'un voltmètre quelconque à mesurer le temps que met l'aiguille pour faire un aller et retour de 0 a 5 volts. En prenant l'adresse &ffff on est sûr d'avoir des 1 sur tout le port d'adresse. Si votre constante de temps n'est pas suffisante vous aurez alors la valeur moyenne de la tension soit 2,5 volts. Il existe, et on n'en parle pas assez, une troisième sorte de bus : le bus de commande. C'est ce bus qui après la distribution des données et l'adressage, donne (pas pour tous les circuits) la dernière autorisation pour qu'une opération se fasse. Il est surtout spécialisé dans la transmission d'interruptions, mode dialogue (handshake) etc.... Voici un petit schéma qui vous fera peut-être comprendre davantage l'importance de ces trois types de bus.



LANGAGE C

Le langage C fut, tout comme le FORTH créé dans les années 70, dans l'une des pièces sombres et mal chauffées des laboratoires BELL, (créateurs de nombreux standards dont par exemple la norme de transmission BELL des modems) par KEN THOMPSON et DENNIS RITCHIE

Le premier développement qui fut à l'origine du C s'appelait le langage B, qui permettait à ces deux joyeux "rigolos" de programmer des mini-systèmes DIGITAL sans avoir à subir les horreurs de l'assembleur.Lorsque DIGITAL sort ses systèmes PDP-9 et 11, les deux ingénieurs décident d'implanter leur logiciel sur ces nouvelles machines, ceci dans un but de portabilité (récupération des sources développés en B) : le langage C était né. Par la suite, le système UNIX sera réécrit en C, et le système d'exploitation sera lié pour le meilleur et pour le pire au langage C. Le langage C est dit de type compilé, c'est-à-dire qu'il génère depuis des listings sources en textes des programmes en code machine. Sa grande portabilité en a fait un des outils de développement privilégié pour des environnements graphiques tels que GEM (on peut par exemple avec une très faible quantité de modification, prendre un programme développé sous GEM sur ATARI 520 ST, et le recompiler sur AMSTRAD-PC), le MACINTOSH d'APPLE, ou même CP/M et MS-DOS.
Le C est un langage structuré, c'est-à-dire que chaque opération peut être décomposée en procédures elles-mêmes composées de sous-procédures. Une telle présentation de programme rend la lisibilité des listings bien plus grande, et par conséquent, permet une

plus grande facilité lors de la mise au point des programmes, ceux-ci étant présentés le plus logiquement possible. Plus besoin donc de l'instruction GOTO, terreur de la programmation, qui transforme le moindre programme, le plus petit soit-il , en innommable spaghetti illisible. L'un des autres avantages du C est qu'il permet d'être le plus proche possible de la machine, tout en restant un langage très évolué : il est possible de manipuler des BITS, des registres, d'effectuer des sauts dans le système d'exploitation ou à des routines en langage machine. Il n'existe plus en C l'éternel problème de limitation des langages tels que le BASIC par exemple : vous pouvez aussi bien mélanger du code en C que du code machine. La présentation des programmes écrits en C est claire, et décomposée en fonctions et sous-fonctions, dont l'origine est une fonction principale intitulée MAIN. Il est possible de réaliser des boucles (instructions WHILE/WEND, FOR, DO) et des branchements conditionnels (instructions IF, ELSE). Vous pouvez également utiliser des opérateurs logiques tels que == pour signifier une égalité, != pour une inégalité, les signes >,<,<=,>=, ainsi que II pour le OU logique, && pour le ET logique, et ! pour la négation.
Bien sûr, si vous êtes un habitué du BASIC, l'apprentissage du C peut vous paraître long et fastidieux, mais vous verrez vite que ce langage est incontestablement l'un de ceux qui peupleront le paysage informatique de l'avenir, grâce à sa personnalité, et surtout à son excellente portabilité qui permettront de faire fonctionner des programmes sur des machines indifférentes, ceci sans problème majeur de syntaxe.

COMMANDE

S'il existe un univers dans lequel les commandes poussent comme les pâquerettes sur le champs de mars, c'est bien celui de l'informatique. On trouve des commandes partout, dans les systèmes d'exploitation, les langages, les programmes. Mais qu'est-ce donc qu'une commande ?

Tout d'abord, c'est ce que redoute chaque néophyte pénétrant dans la jungle de l'informatique : il s'agit d'un mot, dont la signification se rapproche le plus possible de l'action qu'elle va produire (malheureusement, chers petits Français, l'informatique étant par excellence un monde dominé par les technologies anglo-saxonnes, les commandes sont les trois-quarts du temps exprimées en anglais, sauf dans le LSE. Mais qui connaît le LSE ?).
Une commande peut être accompagnée de paramètres, c'est-à-dire d'une suite de valeurs chiffrées ou textuelles, qui permettront de situer précisément le contexte de l'action.
Il existe des commandes pour tous types d'opérations : le graphisme, la gestion d'une unité de disquette, de mémoire, d'un écran, etc. Le minitel fonctionne avec des mots de commandes, pompeusement appelés mnémoniques. Chaque langage a ses propres commandes.
Par exemple, sous le système d'exploitation CP/M (ou bien CP/M+ ainsi que CP/M 68K, CP/M 86 et DOS Plus), la commande PIP permet à l'utilisateur de copier un fichier. Cette commande peut être suivie d'un nombre incroyable de paramètres, qui peuvent concerner la copie d'un fichier avec archive, sur une unité logique ou physique comme par exemple un port série ou parallèle, la copie avec affichage des noms de fichiers à l'écran, avec affichage du contenu des fichiers à l'écran, avec option d'effacement, avec traduction des majuscules en minuscules dans un fichier de texte et inversement, avec ajout de numéros de lignes sur un fichier de texte, en ignorant un certain code de caractères, etc., etc.
Au début de l'informatique, les commandes étaient simples, mais la demande des utilisateurs aidant, elles se transformèrent en commandes de plus en plus complexes, jusqu'à atteindre un niveau de difficulté à la limite du tolérable. Que penser d'une commande PIP par exemple ayant l'aspect suivant :
PIP A:=B: xx [C F A D3 L I G12 NP6]
Bien sûr, j'ai ici choisi délibérément une commande complexe que vous ne retrouverez peut-être jamais, mais cet exemple est significatif.
Un autre problème dans la syntaxe des commandes concerne leur manque de normalisation, et ceci à tous les niveaux : l'exemple simple de la commande PIP peut se retrouver ici. Sous CP/M, la copie de fichier s'effectue avec une ligne ayant la forme suivante :
PIP A:=B:TEXTE.TXT
Elle aura pour effet de transférer le fichier TEXTE.TXT de l'unité B vers l'unité A.
Si nous voulons reproduire la même chose sous MS-DOS, nous devrons taper la ligne suivante :
COPY B:TEXTE.TXT A:
Comme vous le constatez, la syntaxe n'est pas exactement la même pour CP/M (carac-

tère ":=") et pour MS-DOS (":"), mais plus grave encore, l'ordre du fichier source et du fichier destination n'est pas identique (il semble d'ailleurs être plus rationnel sous MS-DOS que sous DOS Plus). Voici le problème crucial des commandes. Et ceci se reproduit dans tous les cas, quels que soient les systèmes d'exploitation (MS-DOS, CPM, UNIX). Chaque concepteur veut disposer sa touche personnelle, et tous les utilisateurs s'arrachent les cheveux car ils doivent réapprendre à chaque fois la syntaxe de toutes les commandes.
En BASIC, le problème est similaire. Le BASIC n'est pas standardisé, et a évolué trop vite. Excepté le MICROSOFT, qui est le plus répandu, et qui en raison de sa source unique, la société MICROSOFT - est toujours le même, les commandes ne sont pas toujours identiques ; par exemple, SOUND sur AMSTRAD qui sert à gérer une interface musicale, peut devenir PLAY sur le SINCLAIR SPECTRUM. Et ceci se reproduit pour pratiquement tous les langages de programmation, excepté lorsqu'une tentative de normalisation est intervenue, cas du langage C et du FORTH.

Le FORTH qui devenait vraiment trop dispersé, et qui en fait perdait totalement son âme, a subi deux tentatives de normalisation, celle du FIG FORTH, et celle du FORTH 79, qui ont pour but dans chacun des cas de faire en sorte que les commandes de ces langages restent figées, et n'évoluent que lorsque le standard lui-même le réclame. Ainsi, les commandes deviennent identiques, et vous pouvez recompiler un source en FORTH, sur plusieurs machines de nature différentes.

Néanmoins, le problème des commandes reste crucial, et demeure l'un des motifs limitant la pénétration de l'informatique dans certains milieux. C'est certainement pour cela que la tendance actuelle veut que l'on supprime les commandes pour des options plus proches des utilisateurs (souris, icônes, fenêtres, etc.). Les environnements graphiques tels que WINDOWS, GEM ou le système à icônes du MACINTOSH d'APPLE sont nés parce que les commandes sont trop complexes à utiliser, et parce qu'elles sont trop loin des utilisateurs.
Un jour, peut-être, les machines répondront à la parole, et aux gestes, et il ne sera plus nécessaire d'utiliser de commandes.

COMPATIBILITÉ ENTRE LES CPC

Les trois modèles de CPC ne sont pas totalement compatibles en basic ni en adresses mémoires. Si par exemple, vous possédez un CPC 464 et que vous tapez un programme qui est prévu pour le CPC 6128 utilisant certaines fonctions basic telles que FILL ou FRAME, la seule solution sera de réutiliser les fonctions RSX que nous vous proposons. Nous essayerons ici d'analyser ce problème de compatibilité, et si possible, de trouver quelques remèdes.

Le basic

Le basic du CPC664 ainsi que celui du CPC 6128 est plus complet que celui du CPC 464. Outre les fonctions propres au lecteur de disquettes, ils possèdent plus de fonctions graphiques. Il s'agit des fonctions COPYCHR$, DEC$, CURSOR, FILL, FRAME, GRAPHICS PEN, GRAPHICS PAPER et MASK.
Il existe aussi la fonction DERR qui fournit le dernier numéro d'erreur produit ayant pour origine le lecteur de disquettes. Signalons en plus l'instruction ON BREAK CONT qui n'existe pas non plus sur le CPC 464.
Il existe une instruction basic sur le CPC 464 dont le manuel ne parle pas, il s'agit de la fonction DEC$. Un bug de la ROM impose toutefois une syntaxe un peu particulière : il faut taper par exemple A$=DEC$ (A, # "## ,) au lieu de A$ = DEC$ (A, "#,´##"). Il suffit simplement d'ouvrir deux parenthèses et de n'en fermer qu'une.
Cette différence de fonctions basic n'est pas trop génante lorsque vous recopiez un programme. Le problème est exactement le même que lorsque vous adaptez un programme qui est prévu à l'origine pour un autre ordinateur que l'AMSTRAD car les instructions basic ne sont jamais exactement identiques sur tous les micro-ordinateurs.

Le langage machine

Le cas est ici nettement plus complexe. Certains programmes du commerce prévus pour le CPC 464 ne

tournent pas sur les autres CPC car certaines adresses mémoires sont différentes d'un modèle à l'autre. Cela est du en partie au lecteur de disquettes qui occupe une place en RAM non négligeable, mais la véritable cause réside dans les espaces ROM des instructions BASIC supplémentaires du 664 et du 6128. L'exemple le plus frappant est celui des vecteurs mathématiques et des vecteurs graphiques.

C'est la raison pour laquelle, lorsque vous tapez un programme publié dans une revue d'informatique (AM-MAG par exemple) réalisé pour un autre CPC que votre modèle, des problèmes pourront surgir.

Par exemple, lorsque vous avez, dans un programme en langage machine, l'instruction CALL &C37D pour le CPC 6128, qui est l'adresse d'une routine d'affichage, ne fonctionnera pas comme prévu sur les autres CPC dont les adresses correctes sont : &C337 pour le 464 et &C380 pour le 664.
Ceux d'entre vous qui possèdent un CPC 464 et qui désirent rendre compatibles les programmes basic prévus à l'origine pour des CPC 464 ou 6128, devront connaître un minimum le langage machine ainsi que certaines adresses mémoires nécessaires. Certains ouvrages donnent des tables d'adresses mémoires utiles pour AMSTRAD citées à la fin de cet article.

Compatibilité hardware (interfaçage)

Certaines extensions ne sont pas connectables sur tous les CPC comme par exemple le premier synthétiseur vocal fabriqué par AMSTRAD qu'il était impossible de connecter sur le 6128 Car son connecteur était trop grand et n'était pas assez profond pour celui de ce CPC. Vérifiez donc, lorsque vous achetez une extension pour votre AMSTRAD qu'elle est bien compatible avec votre modèle. Les bus de sorties ne sont pas identiques sur tous les modèles de CPC. Exemple : le port d'extension du CPC 6128 n'existe pas sur le CPC 464 sous le même nom, il s'agit ici du port appelé FLOPPY DISC.

Un autre problème de compatibilité qui est en même temps logiciel et matériel est celui de la banque mémoire supplémentaire du CPC 6128 (les CPC 6128 possèdent 128 Ko de RAM contre 64 Ko pour les autres CPC). Il est donc évident que les instructions basic qui utilisent cette mémoire supplémentaire du 6218 ne peuvent être transposées sur les 464 ou 664. Il s'agit des instructions BANKFIND, BANKOPEN, BANKREAD, BANKWRITE, SCREENSWAP et SCREENCOPY. C'est aussi la raison pour laquelle, seul le CPC 6128 tourne sous CP/M 3.0 ou CP/M+. Voici par exemple quelques programmes sur CPC 464 qui pourront augmenter sensiblement la compatibilité avec les autres CPC :

```
org # 9000
tampon defs 4 :
  définition d'une zone de 4
octets pour ld hl, tampon hl
  ld bc, comext
  call #bcd1  appel rsx
  ret
comext dfw table : entrée de
la table de noms des nouvelles fonctions
  jp clut
  jp fram
  jp fill
  jp cuon
  jp coff
  jp mask
table defm CLINPU : appel de
la routine par clinput  defb "T"
+ #80
  defm : appel de la routine
par frame
  defb "E"+ #80
  defm FIL : appel de la
routine par fill
  defb "L"+ #80
  defm CURSON : appel de la
routine par curson
  defb "N"+ #80
  defm CURSOF : appel de la
routine par cursoff
  defb "F"+ #80
  defmMAS :  appel de la
routine par mask
  defb "K"+ #80
  defb#00 : fin de la table des
noms
clut ret nz : retour si on a un
paramètre
call #bb03 : reset du
gestionnaire clavier
  ret : retour basic
fram ret nz : retour si on a un
paramètre
  call #bd19 : évite tremblement de l'écran lors d'un
dessin
  ret : retour au basic
fill cp #5 : y-a-t'il 5 paramètres?
  ret nz : si non retour au basic
  ld a, (ix + 0) : charge couleur
d'encre pour en obtenir le
masque
  call #bc2c
  ld h, (ix + 8) : numéro de la
colonne de gauche dans h
  ld d, (ix + 6) : numéro de la
colonne de droite dans d
  ld l, (ix + 4) : numéro de la
ligne supérieure dans l
  ld e, (ix + 2) : numéro de la
ligne inférieure dans e
  call #bc44 : remplissage de
la figure paramétrée
  ret : retour au basic
cuon ld a, 0 : mettre 0 dans la
variable système
  ld (#b28d), a : pour autoriser
affichage curseur
  ret
coff ld a, #ff : mettre 255 dans
la variable système
  ld (#b28d), a : pour interdire
affichage curseur.
  ret : retour au basic
mask cp #1 : y-a-t'il un paramètre ?
  ret nz : si non retour au basic
  ld a, (ix + 0) : mettre 0 à 255
dans la variable
  ld (#b338), a : système pour
avoir un pointillé plus ou
moins grand.
ret   : retour au basic.
```

commentaires

clinput, frame, curson et cursoff n'ont pas besoin de paramètre. On signalera cependant que clinput n'assure pas totalement la fonction de CLEAR INPUT puisqu'elle efface également les KEY DEF.
Fill ne remplit pas toutes les surfaces présentes sur l'écran mais uniquement celles que vous définissez : fill, point gauche, point haut, point bas, couleur. Il est impératif de mettre ces 5 paramètres sinon le programme vous rendra la main.
Pour mask il suffit de mettre un paramètre entre 0 et 255 pour obtenir le pointillé de votre choix. O donnant un pointillé tous les 8 points et 255 donnant une ligne pleine. Par exemple mask, 15 : plot 200, 200 : draw 400,200 vous donnera une ligne pointillé commençant avec un blanc car vous avez remarqué que 15 en binaire fait 00001111 ; le blanc étant dû à la valeur nulle des bits 4 à 7. A vous de rechercher des compromis originaux.

COMPILATEURS BASIC

Ainsi que la plupart des Basics fournis avec les micro-ordinateurs, le Basic Locomotive qui accompagne les AMSTRAD CPC est un langage interprété, forme correspondant d'ailleurs à la conception originelle du Basic. En résumant quelque peu, cela signifie que les instructions écrites dans ce langage sont traduites à chaque fois que leur exécution se fait.

L'interpréteur BASIC

L'interpréteur Basic est le programme qui permet le contrôle et l'exécution des ordres que vous donnez en Basic à votre ordinateur. Dans le cas des CPC, ce programme est écrit en Assembleur Z80 et est résident dans une ROM particulière. Cette ROM occupe les adresses entre &C000 et &FFFF de l'espace adresse du Z80. Outre cet espace ROM, l'interpréteur se réserve une place en RAM qu'il utilise pour y stocker des pointeurs qui lui sont propres.
La traduction d'une ligne d'instructions se déroule, en Basic interprété. D'abord, la ligne que vous saisissez au clavier est placée, telle que vous l'avez tapée, dans un premier buffer. Si cette ligne ne comporte pas de numéro, elle est exécutée directement. Sinon, l'interpréteur considère qu'il s'agit d'une partie d'un programme et la stocke dans un autre buffer. Elle est alors étudiée pour voir si elle comporte des mots Basic et, dans l'affirmative, ces derniers sont remplacés par des "tokens" dont chacun est la traduction sur 1 octet du mot Basic rencontré. C'est seulement après cette traduction des mots Basic rencontrés que la ligne est chargée en RAM pour son inclusion dans le programme.
Lors de l'exécution du programme, les "tokens" contenus dans chaque ligne seront utilisés comme pointeur dans une table de la ROM qui contient l'adresse des instructions Basic. L'exécution de chacune des instructions se fera alors simplement par un CALL à l'adresse trouvée dans la table.
Cette façon de traduire les ordres composant un programme Basic présente des avantages certains. D'une part, elle permet la correction directe des ordres erronés ou détectés comme tels par l'interpréteur. Modifier une ligne se fait très simplement en affichant et changeant son contenu affiché à l'écran, voire même en tapant une autre ligne ayant le même numéro. De même, pour annuler une ligne de programme il suffit de taper son numéro sans aucune autre indication.
Ainsi utilisé le Basic est un langage très "convivial", facilement utilisable par des débutants, et avec lequel la mise au point des programmes est chose aisée. Toutefois, le fait que l'interpréteur doive traduire le contenu de chaque ligne à chaque exécution du programme ralentit considérablement la vitesse d'éxécution de ce langage. C'est pourquoi sont apparus des compilateurs Basic qui sont certes plus lourds à manipuler mais rendent l'exécution des programmes nettement plus rapide.

A cause de la lenteur du Basic interprété et parce que ce langage a été, en définitive, utilisé pour des applications plus complexes que celles pour lesquelles il avait été conçu, sont apparus des compilateurs Basic.
Ceux-ci travaillent d'une façon radicalement différente des interpréteurs. En effet un programme n'est plus composé d'une succession de lignes qui sont traduites à chaque

exécution. Il est constitué d'une partie "source" qui n'est que le texte du programme tel que vous l'avez tapé au clavier, et d'une partie "objet" qui est la traduction en langage machine (Assembleur Z80 ou 8080 pour les CPC) des instructions qui composent le "source". La partie "objet" est sous une forme directement exécutable par le micro-ordinateur et est créé une fois pour toutes lors de la "compilation" du programme. Cette partie est d'ailleurs stockée sur support magnétique sous la forme d'un fichier binaire et c'est la partie que vous devez appeler par RUN lors du lancement du logiciel. La partie "source" doit être elle aussi conservée mais elle n'est à utiliser que pour modifier le logiciel (auquel cas vous devez le compiler à nouveau pour créer un nouvel objet).

Cette méthode de traduction se traduit par un fort accroissement de la vitesse d'exécution des programmes, même par rapport à un interpréteur rapide comme celui des CPC. Ceux-ci sont exécutés sous la forme de programmes en .BIN (ou .COM sous CP/M) qui sont en langage machine. Par contre, elle provoque une perte d'une partie de la convivialité propre au Basic. Il est impossible, en cas d'erreur, de corriger immédiatement le programme et de relancer son exécution. Vous devrez obligatoirement charger le programme source, le modifier, puis compiler la version modifiée. Seul le fichier .BIN résultant de la dernière modification correspondra à l'ultime version de votre programme. Mais le gain de vitesse obtenu peut vous permettre de créer des programmes dont l'écriture aurait été impensable en Basic interprété. Les compilateurs sont peu nombreux et souvent sujets à caution. Le dernier en date est "Le Messie" d'Ere Informatique, Il refuse malheureusement bon nombre de fonctions et de commandes relatives aux chaînes de caractères, ce qui limite son champs d'application. Le programme ne doit pas excéder 25 Ko, commentaires non compris.
En revanche, il accepte les fonctions trigonométriques, ce qui n'est pas le cas de beaucoup de ses concurrents. Or, c'est ce type de calculs complexes qui ralentit souvent l'exécution d'un programme. Une routine compilée sera dans ce cas la bienvenue : l'accélération peut aller jusqu'à 50 fois !

CPC

C comme Color Personal Computeur. Ce sigle de trois lettres est apparu avec le 464. Ce premier modèle distribué en France très discrètement dès le mois de septembre 1984, mettra moins de six mois à faire parler de lui en termes élogieux. Plébiscité par la presse, adopté par les utilisateurs, l'ordinateur Amstrad CPC 464 est toujours en vente et toujours avec autant de succès. Ce premier "bébé" signé Amstrad possède une unité centrale, un clavier, un moniteur et un lecteur de cassettes intégré. L'originalité de l'ensemble est surtout sa prise de courant unique pour l'alimentation des différents périphériques. Fini l'ordinateur "centrale électrique", Avec Amstrad débute l'ère du "tout en un". Plus de deux ans après sa sortie, ce modèle de CPC a prouvé toute sa solidité.

Quelques mois après, Amstrad lance le CPC 664. Celui-ci apporte de nouveaux atouts comme le lecteur de disquettes (format 3") intégré, un nouveau clavier et un design général légèrement différent. La mémoire vive reste identique avec 64 Ko disponibles. Le 664 ne restera que le temps d'une saison dans les magasins, il sera (trop) vite remplacé par une nouveau modèle: le CPC 6128. Avec ce dernier, le "look" est encore modifié, la Ram passe à 128 Ko, on retrouve toutefois le lecteur de disquettes.

Les nouveaux acquéreurs de CPC peuvent choisir aujourd'hui le 464 ou le 6128. Le premier offre un prix d'achat très bas, mais ne permet pas l'utilisation de la disquette. Ce n'est pas très grave puisqu'il est possible d'acheter par la suite une ou deux unités 3". Si vous optez pour ce modèle sachez que le premier lecteur de disquettes devra, en général, être de format 3". Par contre le second pourra avoir un format de 3 1/2" ou de 5 1/4". Ce dernier format offrant les disquettes les moins chères du marché. Avec le 6128, vous pourrez utiliser une seconde unité dans les trois formats, puisque vous possédez déjà une unité intégrée en 3", contenant le contrôleur.

Si nous devions vous aidez dans l'achat d'un CPC, nous dirions que le 464 nous semble l'ordinateur idéal pour ceux qui ne souhaitent qu'une utilisation ludique. On trouve la plupart des jeux sur cassettes et le choix est très important. Si l'on veut une utilisation polyvalente, ludique et gestion personnelle, le 6128 est le meilleur des deux. Il possède une unité de disquettes d'origine et supporte des logiciels comme Dbase II, Wordstar ou Multiplan (notez

qu'il s'agit de produits adaptés aux possibilités de la machine). Alors entre le 6128 et 464, à vous de faire le choix. Ne vous inquiétez pas pour autant, vous ne serez jamais vraiment perdant.
Fiches techniques des modèles de CPC (on trouve facilement des 664 d'occasion)

* *
* * * * * * * * *
* *

Fiche technique du CPC 464

Microprocesseur: Z80A à 4 Mhz.
Rom (mémoire morte) : 32 Ko.
Ram (mémoire vive) : 64 Ko.
Mémoire de masse : Lecteur/enregistreur de cassettes intégré.
Système d'exploitation : Amsdos (CP/M 2.2 en option, livré avec l'unité de disquettes DDI-1).
Interfaces d'origine : Prise joystick, imprimante (Centronics), unité de disquettes, sortie son.
Langage : Basic (Logo en option, livré avec le DDI-1).
Graphisme : Monochrome ou couleur.
Mode 0 : 25 lignes de 40 colonnes, 160 X 200 points, 16 couleurs parmi 27.
Mode 1 : 25 lignes de 40 colonnes, 320 X 200 points, 4 couleurs parmi 27.
Mode 2 : 25 lignes de 80 colonnes, 640 X 200 points, 2 couleurs parmi 27.
Clavier : Qwerty de 74 touches. Pavé numérique programmable.
Option : Unité de disquettes DDI-1 (format Hitachi 3 pouces), imprimante, joystick, Péritel, RS 232, souris, tablette graphique, etc.
Documentation : Manuel complet sur l'utilisation de la machine et la programmation du Basic. Manuel supplémentaire avec l'unité de disquettes.
Assistance : Service technique Amsoft (Sèvres) et Service Après Vente (France entière).

*

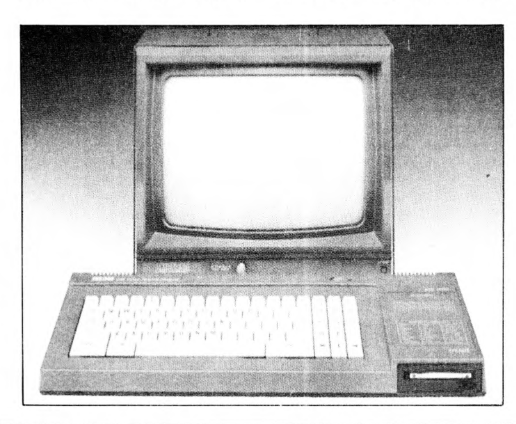
Fiche technique du CPC 664

Microprocesseur : Z80A à 4 Mhz.
Rom (mémoire morte) : 32 Ko.
Ram (mémoire vive) : 64 Ko.
Mémoire de masse : Unité de disquettes 3" intégrée (360 Ko formatés).
Systèmes d'exploitation: Amsdos, CP/M 2.2.
Interfaces d'origine : Prise joystick, imprimante (Centronics), lecteur de cassettes, unité de disquettes supplémentaire, sortie son, périphériques divers.
Langages : Basic, Logo (sous CP/M).
Graphisme : Monochrome ou couleur.
Mode 0 : 25 lignes de 40 colonnes, 160 X 200 points, 16 couleurs parmi 27.
Mode 1 : 25 lignes de 40 colonnes, 320 X 200 points, 4 couleurs parmi 27.
Mode 2 : 25 lignes de 80 colonnes, 640 X 200 points, 2 couleurs parmi 27.
Clavier : Qwerty de 74 touches. Pavé numérique programmable.
Option : Unité de disquettes FD-1, imprimante, joystick, Péritel, RS 232, souris, tablette graphique, lecteur de cassettes, etc.
Documentation : Manuel complet sur l'utilisation de la machine et la programmation du Basic.
Assistance : Service technique Amsoft (Sèvres) et Service Après Vente (France entière).



Fiche technique du CPC 6128

Microprocesseur : Z80A à 4 Mhz.
Rom (mémoire morte) : 48 Ko.
Ram (mémoire vive) : 128 Ko (dont 64 Ko gérés par "Bank Manager").
Mémoire de masse : Unité de disquettes 3" intégrée (360 Ko formatés).
Systèmes d'exploitation: Amsdos, CP/M 2.2., CP/M

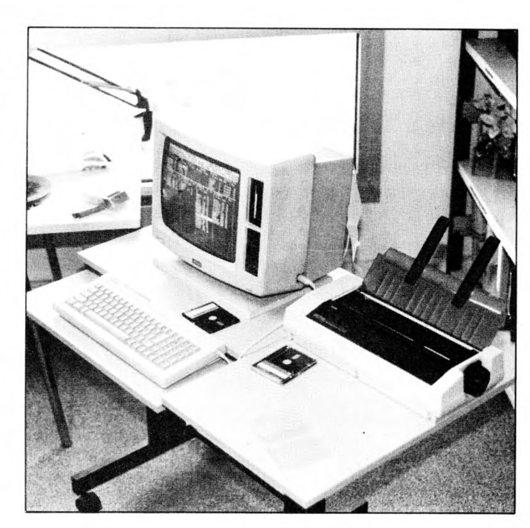


Plus.
Interfaces d'origine : Prise joystick, imprimante (Centronics), lecteur de cassettes, unité de disquettes supplémentaire, sortie son, périphériques divers.
Langages : Basic, Logo (sous CP/M).
Graphisme : Monochrome ou couleur.
Mode 0 : 25 lignes de 40 colonnes, 160 X 200 points, 16 couleurs parmi 27.
Mode 1 : 25 lignes de 40 colonnes, 320 X 200 points, 4 couleurs parmi 27.
Mode 2 : 25 lignes de 80 colonnes, 640 X 200 points, 2 couleurs parmi 27.
Clavier : Qwerty de 74 touches. Pavé numérique programmable.
Option : Unité de disquettes FD-1, imprimante, joystick, Péritel, RS 232, souris, tablette graphique, lecteur de cassettes, etc.
Documentation: Manuel complet sur l'utilisation de la machine et la programmation du Basic.
Assistance : Service technique Amsoft (Sèvres) et Service Après Vente (France entière).



Fiche technique du PCW 8256.

Microprocesseur : Z80A à 8 Mhz.
Rom (mémoire morte) : 256 octets.
Ram (mémoire vive) : 256 Ko, extensibles à 512 Ko.
Mémoire de masse: Unité de disquettes 3" (360 Ko formatés). En option, second lecteur 3" (720 Ko formatés).
Système d'exploitation : CP/M Plus.
Langage : Basic Mallard, Logo.
Logiciel : Traitement de textes "Locoscript" (Locomotive Software).
Graphisme : Monochrome (noir et vert), 32 lignes de 90 colonnes.



Clavier : Azerty de 82 touches (plusieurs touches dédiées au traitement de textes et à la gestion de l'imprimante).
Option : Souris, tablette graphique, lecteur de disquettes, disque dur, etc.
Documentation : Deux manuels complets sur la machine, la programmation du Basic et sur l'utilisation de Locoscript.
Assistance : Service technique Amsoft (Sèvres) et Service Après Vente (France entière).



Parmi les systèmes d'exploitation les plus connus, on retrouve bien entendu CP/M. Celui-ci est très ancien, et se trouve donc de moins en moins utilisé, mais il est aussi le plus prestigieux parce qu'il fut le premier.

Créé par Gary KILDALL aux environs des années 1976, ce système d'exploitation était conçu dans le but de rendre les micro-ordinateurs souples d'utilisation et évolutifs. CP/M fut commercialisé par la société DIGITAL RESEARCH, dont Gary KILDALL est le fondateur, et c'est incontestablement grâce à ce système que cette société américaine acquit ses lettres de noblesse.
Les principaux atouts de CP/M sont d'abord sa portabilité.

Un logiciel développé sous CP/M sur une machine X est tout-à-fait à même de tourner sur une machine Y équipée de ce même système. CP/M ne tournait à l'origine que sur les micro-ordinateurs équipés du processeur 8080 d'INTEL, mais fut en mesure par la suite de fonctionner en mode d'émulation, sur les ordinateurs à base de microprocesseur Z 80 (ce processeur ayant des noms de mnémoniques différents, mais conservant une compatibilité ascendante avec le 8080 au niveau du code machine, par exemple l'instruction de chargement de registre s'écrit LD sous Z80 et MOV sous 8080, mais le code généré après assemblage a la même valeur hexadécimale).

CP/M était prévu, et à l'époque ce fut une innovation, pour gérer efficacement tous les périphériques d'entrées-sorties que comporte un ordinateur : lecteur de cartes perforées (très répandus dans les années 1970), mémoires de masse (disque 8" le plus souvent au début, puis le format 5" 1/4, ainsi que divers types de disques durs), interfaces série ou parallèle pour les imprimantes, unité console (à cette époque, un micro-ordinateur était constitué d'une unité centrale, et il fallait lui relier un terminal qui servait de contrôle pour la visualisation).

CP/M inclut des fonctionnalités dans la gestion des fichiers. PIP permet par exemple de copier un élément d'une unité sur une autre, l'unité en question pouvant être un lecteur de disquettes, une interface logique (canal d'impression, interface série, écran, clavier etc.). D'autres instructions telles DIR permettent d'obtenir le contenu de la disquette, STAT ou SHOW peuvent vous retourner des informations sur le système et sur la configuration de la machine. DEVICE permet de configurer les canaux d'entrées sorties, FORMAT peut formater des disquettes. L'une des autres innovations de CPM, était la possibilité de traiter une application intégralement à l'aide de fichiers de commandes. Ces fichiers de commandes repris ensuite par MS-DOS (les fameux fichiers AUTOEXEC. BAT et tous les autres fichiers se terminant par BAT), donnent à l'utilisateur la possibilité de réaliser des mini-programmes utilisant les commandes du DOS. Ils ont tous les format .SUB, et sont exécutés par le logiciel utilitaire SUBMIT. COM. Ces fichiers peuvent être appelés simplement en tapant une commande au clavier de type :

SUBMIT FICHIER. SUB

Si le fichier en question est intitulé PROFILE ; SUB, il sera pris en compte par le système dès l'initialisation de la machine, si le programme SUBMIT se trouve sur la disquette. Il est possible de placer dans ces fichiers, qui sont des fichiers de texte, toutes les commandes internes ou externes de CP/M (DIR, ERA, DEVICE, PIP). Vous pouvez aussi appeler depuis un fichier SUB un autre fichier de commandes en introduisant une ligne de type SUBMIT, faisant appel au sous-fichier, dans le fichier à exécuter. Comme vous le voyez, le CP/M est loin d'être limité, et sa puissance parfois trop grande, aurait un peu tendance à le rendre complexe à nos yeux, en ces années 86 où l'informatique a bien évolué. CP/M est également équipé d'un éditeur de texte et d'un debugger, tout comme dans MS-DOS (qui a pris l'idée à l'autre...). L'éditeur s'intitule ED, et est aussi lamentable à utiliser qu'EDLIN, et le debugger se nomme DDT pour CPM/80, et SID dans CPM 3.0).

Car ici ntervient une autre des particularités de CP/M, c'est son évolution au fil des années. Quand CP/M 80, qui avec sa zone mémoire limitée à 64 K pour le BIOS et le TPA parut bien faible, DIGITAL introduisit le CP/M 3.0. Celui-ci offrait à l'utilisateur non seulement la possibilité de gérer un TPA de 61 K maximum (TPA signifie Transient Program Area, c'est-à-dire la zone maximum dans laquelle peut s'exécuter un logiciel), mais permettait de travailler en couleurs (quatre au maximum) et, à l'aide d'une seconde couche du système d'exploitation, de travailler avec un véritable environnement graphique intitulé GSX. Ce système, lié à un logiciel quelconque, implantait en mémoire divers vecteurs pour des procédures graphiques de base (primitives) telles que l'affichage d'un point à l'écran, le tracé d'une ligne, d'un polygône etc.

Après avoir introduit le concept de portabilité des logiciels, DIGITAL offrait aux utilisateurs la possibilité de transférer aussi les graphismes et les logiciels de dessins (DRDRAW et DRGRAPH).

Malheureusement, CP/M 3.0 intitulé également CP/M + fut un échec à une époque où MICROSOFT n'était pas encore le géant que l'on connaît aujourd'hui. Une tentative fut faite pour adapter CP/M aux machines à base de processeur 8088 ou 8086, afin de contrer MS-DOS, qui avait déjà été choisi par IBM pour équiper ses PC, mais il était déjà trop tard, et CP/M 86, même s'il eut un petit succès (c'est le second système d'exploitation sur PC après MS-DOS), ne put jamais égaler MS-DOS. Pourtant, les logiciels développés sous CP/M 80 ou CP/M 3.0 sont facilement transportables (seuls les sources sont à transplanter et à recompiler sur CP/M 86) et l'imposante bibliothèque de logiciels disponibles sur systèmes CP/M aurait pu être un atout de choix pour CP/M 86 mais les petits plus de MS-DOS ont suffit pour faire la différence.

CRTC 6845

Le CRTC 6845 est un circuit intégré LSI (Large Scale Integration = grand taux d'intégration) générant les signaux nécessaires à la production d'une image vidéo. A partir d'un signal d'horloge unique, il produit tous les signaux nécessaires pour le moniteur. Tous ces paramètres sont programmables.

Voici les principales possibilités de ce circuit:
- curseur programmable (hauteur et clignotement).
- fonctions de contrôle du curseur.
- matrice de points des caractères programmable.
- nombre de lignes par écran programmable.
- accès à une zone de 16 ko.
- entrée stylo-optique.

Ce circuit comporte dix-neuf registres internes dont un, le registre adresse, est utilisé pour sélectionner celui sur lequel on désire travailler. Les autres registres ont appelés registres paramètres. Deux étapes sont nécessaires pour accéder à un registre :

1) Il faut d'abord fournir, au registre adresses, le numéro du registre auquel on désire accéder. Il y a dix-huit registres paramètres, donc il faut cinq bits pour les adresser : donc le registre adresse est un registre 5 bits.

2) Ecrire ou lire dans le registre sélectionné.

Les registres paramètres, numérotés de R0 à R17, peuvent être subdivisés en trois groupes principaux :

1 - R0 à R3 : Programmation du format horizontal.

2 - R4 à R9 : Programmation du format vertical.

3 - R10 à R17 : contrôle du curseur, de la RAM et du stylo-optique.

Ces registres sont en général à écriture seule, à l'exception des deux registres de contrôle de la position du stylo-optique qui sont, bien sûr à lecteur seule, et des deux registres du contrôle de la position du curseur qui peuvent être utilisés en lecteur ou en écriture.

Fonctions de registres du CRTC 6845

Registre 0 : Nb de caractères total en horizontal 0-255.
Registre 1 : Nb de caractères affichés en horizontal 0-255.
Registre 2 : Position de la synchronisation horizontale 0-255.
Registre 3 : Longueur de la synchronisation 0-15.
Registre 4 : Nb total de lignes en vertical 0-127.
Registre 5 : Synchronisation verticale 0-127.
Registre 6 : Nb de lignes réelles affichées 0-127.
Registre 7 : Position de la synchronisation verticale 0-127.
Registre 8 : Mode entrelacé 0-3.
Registre 9 : Nb de lignes par écran 0-31.
Registre10 : Position et forme du curseur 0-31.
Registre 11 : Ligne de fin de curseur 0-31.
Registre 12 : Adresse haute début RAM écran 0-63.
Registre13 : Adresse basse début RAM écran 0-255.
Registre 14 : Adresse haute de la position du curseur 0-63.
Registre 15 : Adresse basse de la position du curseur 0-255.
Registre 16 : Adresse haute de la position du stylo-optique 0-63.
Registre 17 : Adresse basse de la position du sytlo-optique 0-255.

Voyons plus en détail l'utilisation du registre 10 : les bits 0 à 4 déterminent sur quelle ligne de la grille doit commencer le curseur. Les bits 5 et 6 fixent le curseur de la manière suivante :

BITS

6	5	
0	0	Curseur non clignotant.
0	1	Curseur non représenté.
1	0	Curseur clignotant (environ 3 fois par seconde).
1	1	Curseur clignotant (environ 1.5 fois par seconde).

programmation du CRTC 6845 sur les CPC

Quatre adresses de port sont nécessaires pour programmer le CRTC :

Le port &BCXX sert à écrire dans le registre adresse, donc à sélectionner le registre auquel on désire accéder.

Le port &BDXX qui sert à écrire dans le registre sélectionné.

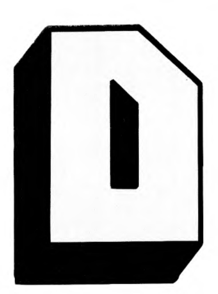
Le port &BEXX, utilisé en lecture uniquement, sert à lire le contenu du registre d'état du CRTC.

Le port &BFXX sert à lire le contenu du registre sélectionné auparavant par le port &BCXX. Seuls, les registres 14, 15, 16 et 17 peuvent être lus.

Donc, pour écrire 39 dans le registre 1 en BASIC, il suffit de faire : OUT &BC00, 1 : OUT &BD00,39.

ATTENTION, certaines valeurs peuvent "planter" l'ordinateur.

La programmation de ce circuit sera utile à tous ceux qui recherchent des effets un peu particuliers sur l'écran de leur cher AMSTRAD. Amusez-vous donc avec quelques valeurs, vous ne risquez pas de détériorer votre CPC. Tout ce qui pourrait vous arriver, serait d'obtenir des résultats bizarres !



DATA

Les datas sont des données stockées sous diverses formes dans un programme ou dans un fichier pour être utilisées ultérieurement par un logiciel. Les data peuvent être présentées sous plusieurs formes : textuelle, binaire, hexadécimale, décimale, ou mixées dans tous ces types de formats.

Lorsqu'une entrée de données peut s'avérer pénible, par exemple à l'aide d'une suite d'instructions INPUT, il est préférable que ces données puissent être figées, pour les stocker dans une suite de lignes de DATA dans un programme. Vous pouvez par exemple dans un programme basic utiliser la commande DATA pour stocker des constantes (valeurs inaltérables) qui seront lues dans un ordre séquentiel (les unes après les autres) par une commande "read" du BASIC. Au fur et à mesure que chaque valeur est lue, un pointeur passe à la valeur suivante, et ce jusqu'à ce que la liste soit épuisée. Il est possible de "restorer" un point de lecteur de data dans un programme basic avec la commande RESTORE. Vous pouvez dans un programme BASIC mixer des données derrière un ordre data comme par exemple : ·

DATA "hello", 10, "Bonjour", 20.

Ce qui nous donnera en situation réelle :

10 READ, A,B,C : REM lecture des variables a,b,c.

20 PRINT A,B,C : REM affichage des données lues.

30 DATA 10,20, 30 : REM endroit de stockage des données.

40 STOP.

En règle générale, les données alphabétiques (ASCII) sont placées entre guillemets, les données numériques hexa-décimales suivies par un H, les données numériques binaires précédées ou suivies par un B, les données décimales sont indiquées par leur valeur décimale sans indicateur. Dans un programme en assembleur, les data peuvent être stockées derrière une instruction de type DB (Define Byte ce qui veut dire réserver des espaces d'un octet), DW (Define Word ce qui veut dire réserver un espace de deux octets), DM qui annonce qu'une chaîne ASCII (chaîne) de texte va être stockée. Les data ne sont pas nécessairement conservées dans un programme, on parle également de data dans le cas de fichier de texte pour un traitement de texte, un tableur, ou un intégré. Ces fichiers sont alors conservés sur disquette. En fait, il ne faut pas considérer que l'on parle de data uniquement dans le cadre d'un logiciel en basic qui utilise ce type d'instructions. Tout logiciel utilise des data, ne serait-ce que pour afficher du texte ou du graphisme, seule la forme de stockage diffère d'une application à une autre, voire d'une machine à une autre.

DISQUETTES

Vous avez tous déjà entendu parler des disquettes. Les heureux possesseurs du CPC 664, CPC 6128 ou CPC 464 + DD1 en utilisent très fréquemment et connaissent leurs avantages par rapport aux cassettes. Nous allons ici essayer de vous les décrire très simplement.

Vous savez tous que les CPC, comme tous les ordinateurs, ont une mémoire volatile, c'est-à-dire qu'ils perdent leurs données dès qu'ils ne sont plus sous tension. Les cassettes ou les disquettes ont été conçues pour remédier à ce problème. Imaginez que vous ne puissiez sauvegarder vos programmes ! Vous devriez alors retaper entièrement le programme dont vous avez besoin ! Les cassettes et les disquettes sont là pour éviter cette peine. Il y a encore quelques années, le fait de posséder un micro-ordinateur avec un ou plusieurs lecteurs de disquettes était un luxe. Heureusement, les constructeurs (dont AMSTRAD) se sont efforcés de commercialiser des lecteurs de plus en plus fiables et de moins en moins chers.

Il existe quatre tailles de

disquettes.
- Les 8 pouces sont des fossiles survivants dans des entreprises attachées à leur ancien matériel.
- Les 5 pouces 1/4 ne sont en réalité que des 8 pouces de taille réduite. Elles en conservent les défauts, à savoir une grande fragilité liée à leur souplesse et une fenêtre de lecture ouverte sur les agressions extérieures. Elles sont encore le support privilégié d'un immense parc de PC et compatibles. Leur atout : un prix de revient dérisoire.
- Les 3 pouces 1/2. Arrivées dans le scepticisme général, elles se sont imposées sur des machines prestigieuses (MacIntosh, IBM...) et sur l'ensemble des portables. D'une rigidité à toute épreuve, un volet protège la galette magnétique, d'où une fiabilité et une sécurité exemplaires. Malgré leur petite taille, la capacité de stockage est élevée : jusqu'à 1,44 méga-octets ! - Les 3 pouces. Choix malheureux d'Amstrad, qui reste la seule marque à l'utiliser pour la gamme CPC et PCW. Elle reçoit jusqu'à 720 Ko et sa robustesse est légendaire.

Avantages des disquettes sur les cassettes

Le plus important est la rapidité de chargement ou de sauvegarde de données. Sur une cassette, il faut faire défiler la bande et prendre le programme à son passage. D'où une perte de temps en recherche. Sur la disquette, l'ordinateur va chercher ce programme presque directement.
Tous les renseignements sur la position physique de celui-ci sur la disquette lui sont fournis dans le directory (le directory est l'espace de la disquette où se trouvent les noms de tous les programmes contenus sur la face lue de la disquette ainsi que des renseignements pour l'ordinateur sur la longueur, la position des programmes, ...)
Un autre avantage de la disquette est sa fiabilité. Elle est nettement plus grande que celle d'une cassette. Les cassettes que vous achetez dans le commerce ne sont pas prévues pour cette utilisation mais pour de l'enregistrement musical ou vocal. Il existe cependant des cassettes spéciales pour l'informatique qui sont plus chères et dont la durée d'enregistrement est quelquefois plus courte.
De plus, certains d'entre vous utilisant des cassettes se sont certainement heurtés aux problèmes de réglages. Il faut un niveau de volume correct à la sortie du magnétophone ainsi qu'un bon réglage d'azimutage de la tête de lecture (celle-ci doit être bien positionnée par rapport à la bande magnétique).
Il n'y a aucun réglage à faire sur les lecteurs de disquettes, (ils ont été faits en usine par le fabricant) tout ce que vous avez à faire, c'est de formater vos disquettes lorsqu'elles sont vierges.
Attention : ne formatez jamais une disquette contenant des données car celles-ci seraient irrémédiablement perdues ! Le formatage est réalisé par un logiciel fourni avec la machine (DISCKIT par exemple, qui comprend une instruction FORMAT).

Division d'une disquette

Une disquette 3 pouces possède deux faces utilisables indépendamment. Elles peuvent bien sûr être formatées différemment.
On divise une disquette en pistes et en secteurs. Quel que soit le format utilisé, il y aura toujours quarante pistes. Seul, le nombre de secteurs peut varier (8 en format IBM, 9 sinon) qui contiennent 512 octets soit un demi kilo-octet (un kilo-octet, noté Ko, équivaut à 1024 octets). Pour CP/M, il existe une autre subdivision appelée ENREGISTREMENT comprenant 128 octets. Cela est nécessaire pour une raison de compatibilité avec CP/M. Chaque secteur contient donc quatre enregistrements. AMSDOS utilise des BLOCS qui sont constitués de deux secteurs donc 1024 octets. C'est la plus petite unité pouvant être appelée depuis le Basic AMSTRAD.
Les numéros de secteurs dépendent du format utilisé. En format système, ils sont numérotés de &41 à &49, et de &C1 à &C9 en format DATA. Le format système est utilisé pour une utilisation de disquette en CP/M. Lors du formatage l'AMSTRAD copiera sur les deux premières pistes le système d'exporation CP/M. Le format VENDOR est identique au format système en ce qui



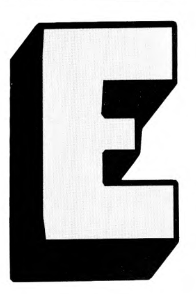
concerne les numéros des secteurs. Les deux premières pistes sont libres mais on ne peut y accéder depuis le Basic. Dans ces deux formats, le directory se situe en piste deux secteurs &41 à &43. Alors qu'en format DATA, il se situe piste zéro, secteurs &C1 à &C3.
Pour de plus amples renseignements, nous vous conseillons le livre du lecteur de disquettes de chez Micro-Application.

Eric MISTELET

DOS PLUS

DOS Plus est le dernier système d'exploitation de DIGITAL RE-SEARCH ? Ceci est archi faux ! DOS Plus a déjà été implanté sur d'autres machines, mais cet événement fit tellement de bruit que personne ne sait quelles sont ces machines. Une chose est sûre, ce Dos a été implanté sur le dernier prototype de la société ACORN, avant que celle-ci, au bord du gouffre, n'ait été rachetée par OLIVETTI ; cette machine ne vit donc jamais le jour.

Mais revenons à cette oppressante question : qu'est-ce que DOS Plus ? DOS Plus est un système d'exploitation pour les ordinateurs à microprocesseurs 8086, 8088. Il reprend d'une certaine façon tout l'acquis de DIGITAL RESEARCH, avec des nouveautés dues à des études marketing poussées, telle que la compatibilité avec MS-DOS version 2. Ce DOS accepte donc des programmes sur format de disquettes CP/M ou MS-DOS, et émule les systèmes d'exploitation CP/M 86, MS-DOS, CONCURRENT CP/M, ainsi que CONCURRENT PC-DOS. Notons toutefois que pour des systèmes multi-utilisateurs tels que CONCURRENT, DOS Plus n'accepte que les logiciels en version mono-utilisateur (faut quand même pas exagérer !).
DOS Plus apporte de nombreuses améliorations par rapport aux DOS courants, puisqu'il permet de travailler indifféremment sur des fichiers de format CP/M ou MS-DOS, ceci sans avoir besoin d'effectuer des manipulations complexes. Il permet également un mode de fonctionnement en simili multi-tâches, avec la notion de trois programmes en arrière-plan et un programme d'avant-plan pouvant fonctionner simultanément. Attention toutefois, les logiciels d'arrière-plan sont des programmes développés spécifiquement inutiles donc de rêver de SUPERCALC fonctionnant en même temps que DBASE et REFLEX. Une ligne de commande en bas de l'écran affiche en permanence des informations sur le système, l'heure, et le ou les logiciels en cours d'exécution.
DOS Plus en mode MS-DOS est, ceci est intéressant, légèrement plus rapide que l'original de MICROSOFT. La compatibilité entre les deux systèmes n'est pas des plus parfaite, des logiciels tels GWBASIC, SIDECKIK ne fonctionnent pas avec DOS Plus, mais les grands classiques tournent (une liste est communiquée par la société AMSTRAD).
Pour finir, sachez que l'AMSTRAD PC n'est plus, depuis quelques temps le seul compatible équipé de DOS Plus, puisque le JASMIN PC le détient en série.



ÉLECTRONIQUE ET ORDINATEURS

Vous n'êtes pas sans savoir que l'informatique s'étend de plus en plus dans la société. Dans l'industrie électronique, l'ordinateur est utilisé pour de nombreuses applications que nous allons ici essayer de vous décrire.

Interfaces

L'ordinateur seul n'a pas de spécialisation précise. Pour le relier au monde extérieur (saisie de données physiques, commandes ...), il lui faut un module d'adaptation pour rendre compatibles ses signaux avec ceux du système de prises de données (capteur par exemple). Cette fonction est réalisée par les interfaces.

Elles permettent de spécialiser l'ordinateur dans une tâche externe bien définie. Celui-ci doit, bien sûr, fonctionner avec un logiciel possédant des routines de lecture et d'écriture de données sur cette interface ainsi qu'une ou plusieurs routines de traitement et de prise de décision. Une interface est spécifique à une application. Sur les CPC 464 par exemple, si vous connectez un lecteur de disquettes DDI-1, vous remarquerez un boîtier se trouvant entre l'AMSTRAD et le lecteur. C'est une interface permettant de convertir des signaux provenant du lecteur de disquettes pour qu'ils soient compatibles en synchronisation, vitesse, pour le CPC, et inversement.
L'interface n'est pas forcément séparée du module d'acquisition de données ou de commandes. Elle peut fort bien être une sous-partie de celle-ci. Par exemple, dans l'imprimante DMP 2000, l'interface se trouve dans celle-ci. En résumé, l'interface est un système de mise en forme de signaux et de transcodage nécessaire à la liaison de l'ordinateur et de ses périphériques.

Les capteurs

Pour prendre connaissance de données extérieures telles la température, ou une position, l'ordinateur acquiert ses données via l'interface depuis ses périphériques. La première étape consiste à associer la grandeur à mesurer à une grandeur électrique qui peut être analogique (elle serait alors convertie en numérique grâce à un convertisseur analogique-numérique). Cette fonction est asssurée par un capteur. C'est un composant qui permet une adaptation entre le système observé et le dispositif de traitement. Dans le cas du traitement par ordinateur, le capteur fournit une grandeur électrique (intensité ou différence de potentiel) qui est proportionnelle ou inversement proportionnelle à la grandeur mesurée.
Exemple de capteurs : microphones, caméra vidéo, thermistances, photodiodes, potentiomètres, ils associent une valeur chimique, ou résistance électrique, à une position d'un mobile)...

Lors de la fabrication robotisée, on utilise la reconnaissance de formes. Cela est utile lorsque le robot prend une pièce. Celle-ci n'est pas obligatoirement disposée comme il faut sur la table. Le robot doit donc la reconnaître, la rejeter si elle présente un défaut et la manipuler pour la positionner dans le bon sens pour pouvoir ensuite la monter comme prévu lors de l'assemblage. Un robot de transport de matériel doit aussi posséder cette faculté pour pouvoir reconnaître d'éventuels obstacles avant d'être en contact avec eux. Ceci est aussi une des applications possibles de l'ordinateur dans l'industrie.

Systèmes experts

L'une des utilisations les plus courantes pour l'ordinateur est le système expert. En effet, le système expert est un logiciel qui au début de sa création possède des connaissances qu'on lui a données par le biais d'une analyse approfondie de celles qu'utilise habituellement un spécialiste. Le système expert englobe un grand nombre de matières, la CAO, la robotique, qui se subdivisent en d'autres nombreuses catégories. La Fig. 1 montre les types d'outils utilisés dans les différentes applications de conception en électronique. Le type d'outil de CAO utilisé varie selon que l'on travaille sur un circuit analogique ou digital et entre les technologies du circuit imprimé ou des composants électroniques.
Les concepteurs de circuits intégrés utilisent la CAO pour résoudre des problèmes de conception réputés impossibles à résoudre sans aide informatique. Le concepteur de circuits imprimés, d'un autre côté, a un modèle bien meilleur de son oeuvre, possédant une bonne connaissance de ce qui est un bon projet de circuits imprimés. Le concepteur de circuits intégrés ne peut pas physiquement commencer à développer une compétence artisanale de la conception de puces. Par exemple, le principal outil de CAO utilisé par les concepteurs de circuits logiques est le simulateur de circuits logiques.
Le contrôle fonctionnel est un problème beaucoup plus intéressant que la simple vérification. Il a pour but d'assurer que les fonctions logiques impolantées vont, en entrées/sorties, produire en un certain nombre de points d'essais prédéfinis, le comportement prévu et contrôlé. Le format de sortie dans certains cas est sous le contrôle de l'ingénieur qui peut grouper des ensembles de signaux sous une forme significative. Il n'est pas possible de simuler toutes les conditions possibles de fonctionnement du circuit dans les cas pratiques. Ce problème est résolu par l'ingénieur qui conçoit les conditions de testabilité. Un autre exemple important, quant à l'interaction ordinateur-monde extérieur, est la procédure manuelle d'implantation : la conception manuelle des schémas d'implantation est l'un des domaines dans lesquels la CAO a été utilisée pour résoudre la plupart des problèmes inhérents au processus de conception. L'implantation manuelle est logique et requiert beaucoup de dextérité manuelle et de patience : chaque couche d'une plaquette de circuit imprimé, par exemple serait représentée sur une couche séparée du

film transparent, et les composants avec leurs interconnexions le seraient par des symboles autocollants et des rubans de tissu noir. Ce type de documentation était dégradable (le ruban commençait à s'entortiller au bout d'une quinzaine d'essais) et facile à égarer. Pour des circuits intégrés allant jusqu'à dix mille postes, les dessins d'implantation couvraient tous les murs et étaient impossibles à vérifier. La CAO accélère le processus d'implantation en diminuant le temps employé à la correction des erreurs. De plus, elle fournit une représentation du circuit à la fois stable, vérifiable et accessible conjointement par les concepteurs et les programmes.
La fig. 2 représente un tableau qui résume bien les diverses possibilités de l'ordinateur dans l'industrie. La fig. 3 représente un exemple de ce que peut être l'architecture d'un ensemble conseiller.

UTILISATEURS	APPLICATIONS
DIGITAL EQUIP. CORP.	Contrôle les commandes d'ordinateurs V.A.X
"	Analyse les défaillances d'installations informatiques
"	Assistance aux concepteurs de circuits
"	Dépannage de pannes de réseaux informatiques
FAIRCHILD	Diagnostic des chaînes de fabrication de circuits
G.T.E	Maintenance des commutateurs téléphoniques
HEWLETT-PACKARD	Contr. des phases de photolithographie des circuits intégrés
I.B.M	Aide au pilotage du système d'exploitation M.V.S
LOCKEED	Dépannage d'équipements de communication
"	Dépannage d'instruments électroniques de laboratoire
N.C.R	Contrôle des commandes et configuration d'ordinateurs
S.W BELL	Dépannage des lignes téléphoniques
CAMPBELL SOUPS	Dépannage des stérilisateurs
HITACHI	Contrôle de freinage des trains
KAWASAKI STEEL	Détecte les défauts dans le métal
WESTING HOUSE	Mise en valeur du combustible nucléaire
LIVERMORE Nal. Lab.	Règlage d'un spectromètre de masse
MOLECULAR DESIGN. LTD	Identification des molécules
SUNY-STONYBROOK	Planification de synthèse chimique
NASA	Identifié des minéraux à partir d'images provenant des satellites
SCHLUMBERGER	Analyse les données provenant d'un puit de pétrole
HELENA Labs.	Analyse des protéines de sérum
I.C.I	Prévention des maladies des blés d'hiver
U.S ARMY	Optimise le chargement des avions en équipement de frêt
E.P.A	Conseil sur la divulgation d'informations confidentielles

Fig 2

La Robotique

Pour ce qui est de la robotique, nous allons traiter quelques exemples des plus courants. Tout d'abord, l'utilisation de télémanipulateurs permet de reproduire les gestes de l'opérateur à distance. Par exemple, dans les endroits dangereux où les radiations mettraient sa vie en danger (laboratoire de recherches, centrales nucléaires...), c'est une machine élémentaire dont le bras, souvent télescopique, manipule des objets légers, tels des fioles contenant des produits chimiques dans de nombreuses applications de laboratoire.
L'apprentissage manuel d'un robot devient une manipulation de plus en plus courante. Ainsi, l'opérateur exécute lui-même avec une grande précision, chacun des mouvements qui devra être mémorisé et reproduit par la machine. Il définit notamment l'intensité de la pression qui devra être exercée par la main du robot sur les objets dont elle se saisira en cours d'opération. Le travail à la chaîne des robots dans l'industrie automobile réalise notamment la soudure par point dans les chaînes de montage ; ces robots sont désormais capables de mémoriser plusieurs modèles de carrosseries, de manipuler avec rapidité les volumineux pistoles de soudage, de suivre les chaînes en mouvement tout en réalisant les opérations de soudure. La soudure à l'arc est aussi un domaine d'application pour les robots car elle est dangereuse et pénible pour l'homme, notamment du fait des radiations de rayons ultra-violets, des étincelles et de la fumée dégagée par cette opération. Les robots s'en chargent désormais avec une précision supérieure à celle des opérateurs humains.
Le chargement et le déchargement des palettes est une application de la robotique qui intéresse tant les entreprises industrielles que les entreprises commerciales.
Outre la régularité et la précision, elle offre la possibilité de confier au robot plusieurs opérations à réaliser en alter-

nance.
L'ébavurage confié aux robots est un travail monotone et pénible. C'est aussi un travail de précision qui convient parfaitement aux robots.
L'assemblage est une des nouvelles applications de la robotique. Elle laisse entrevoir des possibilités considérables. De nombreuses recherches tant au Japon qu'aux USA et en Europe portent sur l'attribution aux nouvelles générations de robots de capacités de reconnaissance d'objets disposés au hasard en vue de la robotisation des opérations d'assemblage. Les robots de peinture présentent les deux avantages suivants : une parfaite régularité du travail et une réelle économie de peinture résultant directement de la précision avec laquelle les robots effectuent cette tâche.
On peut noter encore une application dans laquelle les Japonais sont très avancés : il s'agit de la vision artificielle. Le robot dessinateur présenté par Natsushita à l'exposition 85 de Tsukuba au Japon est capable de réaliser en 3 minutes le portrait de toute personne placée dans le champ de sa caméra. A une moindre échelle, il est possible de s'initier à la robotique grâce à Computing Experimental, un ensemble de moteurs pas à pas, de pièces détachées et d'équipements en soriel (détecteurs de lumière et de chaleur). Ce kit de Fisher Technik permet de créer des bras robots, des tables traçantes et autres modules. Une interface le relie a l'ordinateur préalablement programmé pour qu'il puisse accomplir diverses taches.

La télécommunication

Deux ordinateurs pourront dialoguer à l'aide de deux émetteurs-récepteurs et deux modems comme l'indique la fig. 4. L'information transmise est de type linéaire série (unité de 0 et de 1). L'ordinateur travaillant en parallèle, il faut une interface convertissant l'information en série (voir fig. 5). Dans les modes de transmission où les erreurs ne peuvent être ni détectées et donc ni corrigées, l'utilisateur final doit prendre ses propres dispositions pour conférer une certaine redondance aux messages qu'il veut transmettre. C'est le cas du télex, où l'usage veut que, par exemple, les sommes en chiffres soient suivies de leur représentation en toutes lettres. Mais heureusement, les modes de transmission moderne en informatique permettent de détecter les erreurs, afin, soit de les corriger, soit de prévenir l'utilisateur final.
Signalons brièvement deux modes de transmission : la procédure asynchrone (ou TTY) et la procédure synchrone qui permet au récepteur de se synchroniser avec l'émetteur grâce à des caractères spéciaux, dits de synchronisation. L'exemple le plus courant d'utilisation de la procédure TTY dans le cadre d'une communication entre micro-ordinateurs est le transfert de fichiers. Celui-ci peut se faire par lignes spé-

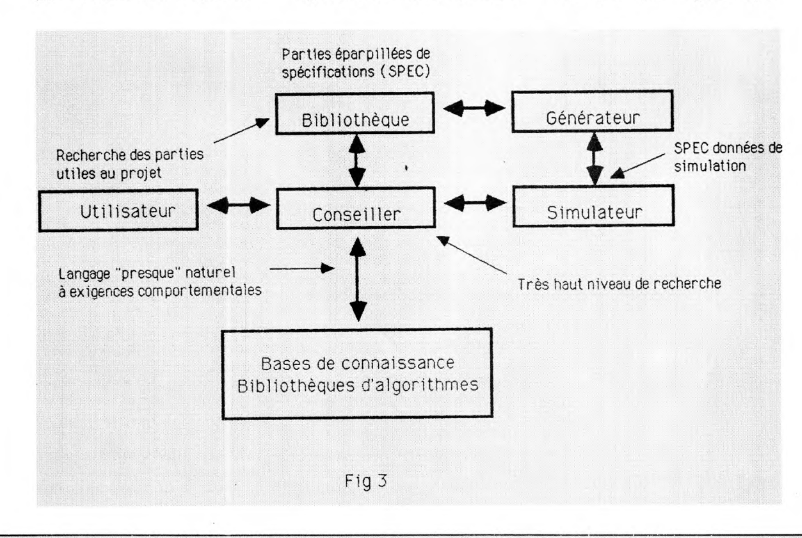


Fig 3

E

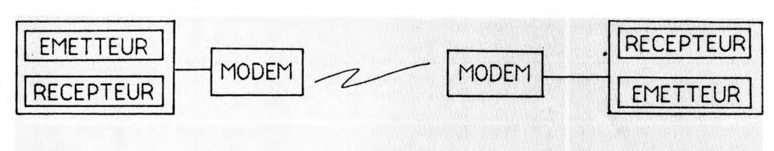
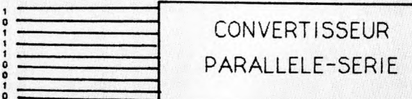


Fig. 4



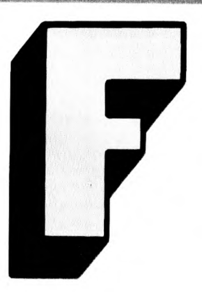
010011101

(Exemple interface RS 232)

Fig. 5

cialisées (LS), par le réseau téléphonique commuté (RTC), ou par l'intermédiaire d'un réseau de communication par paquets de type TRANSPAC. Dans le cas d'un transfert via TRANSPAC, la communication se fait à travers les PAD (Packed Assembler Dessasembler) qui constituent les points d'accès à ce réseau. Ce sont eux qui forment les paquets de données à la réception. La procédure synchrone permet des interactions plus intéressantes. Par exemple, l'interface programme VTI (Virtual Terminal Interface) permet à un programmeur de développer un programme d'application spécifique autour d'un émulateur (une émulation écran permet de recevoir des images écran de l'ordinateur hôte, de les décoder avant de les afficher et d'émettre, après décodage, la saisie de ces écrans vers le site central ; en un mot, l'émulateur écran simule un écran connecté à un ordinateur hôte). Ainsi, à l'aide de l'image virtuelle, le programmeur peut modifier un écran avant l'affichage. Il peut également contrôler la saisie effectuée par l'opérateur, voire la compléter avec des données locales avant d'effectuer une transmission vers le site central.

Eric MISTELET
Guillaume PONTICELLI



FDC/765

Le FDC 765 peut être considéré comme un microprocesseur très spécialisé tant ses possibilités sont étendues. Elles sont d'ailleurs si nombreuses qu'elles n'ont pas toutes été utilisées sur les CPC.

Le format de données utilisé par le FDC correspond au format IBM 3740 en simple densité, et au format IBM system 84 en double densité. De ce fait, les disquettes Commodores ou Apple, par exemple, ne peuvent être lues ni écrites. Ce circuit, de 40 broches, fournit tous les signaux nécessaires pour faire fonctionner l'ensemble des lecteurs de disquettes existant sur le marché, des 8" aux 3". De plus, les signaux de commandes disponibles permettent de connecter le FDC à presque tous les proces-

F

seurs. Voici un aperçu des données techniques de ce circuit :

- longueur de secteur programmable ;
- toutes les données du lecteur programmable ;
- transfert de données aux choix, en mode DMA ou pas en mode DMA ;
- connectable à presque tous les types de processeurs courants ;
- horloge monophase simple de 4 ou 8 MHz.

Le troisième point parle d'un mode DMA, certains d'entre vous se demandent certainement ce que c'est, et nous allons combler cette lacune :
En liaison avec un DMA controller, le FDC a la possibilité de prendre en charge le contrôle de la mémoire du système pour le transfert de données. Il ne nécessite pas de passer par le microprocesseur et il est donc évident que cette méthode de transfert de données est très rapide. Malheureusement ce n'est pas cette méthode qui est utilisée sur le CPC.
Dans notre cas, le transfert est pris en charge par le processeur, qui est un Z80 dans le CPC. Il existe alors deux possibilités d'exploitation du FDC :
La première est la méthode des interruptions qui, étant relativement insatisfaisante, n'a pas été retenue par les développeurs de la carte controller du CPC ; nous n'en parlerons donc pas.
La deuxième, donc celle qui a été choisie, est la méthode polling. Le Z80 examine régulièrement dans les registres du FDC qu'elle est la prochaine action demandée par le FDC.

Programmation du FDC

Le FDC se situe sur les adresses de port &FB7E et &FB7F. A la première adresse se trouve le registre d'état principal, la deuxième appartient au registre de données.
Cependant, une troisième adresse est utilisée par ce circuit, il s'agit du port &FA7E où se trouve une bascule commandant le moteur du lecteur de disquettes. Par souci d'économie certainement, si plusieurs lecteurs de disquettes sont connectés à l'AMSTRAD, ils seront mis en marche et arrêtés ensembles. Si on écrit un 1 sur ce port, les moteurs de ces lecteurs seront mis en marche. Par contre, si on écrit un 0, tous les moteurs seront arrêtés. Nous rappelons qu'en BASIC AMSTRAD, c'est l'instruction OUT qui permet d'écrire sur un port, il vous faudra donc faire un OUT & FA7E, 1 pour mettre en marche les moteurs. La programmation de ce circuit doit être réservée aux spécialistes. Une erreur de manipulation pourrait endommager une disquette se trouvant dans le lecteur. Soyez donc très prudents et ne travaillez qu'avec des disquettes ne contenant rien d'important. Nous vous recommandons le livre du lecteur de disquettes CPC 6128, 664 & 464 de chez MICRO APPLICATION qui répondra à beaucoup des questions que vous vous posez sur tout ce qui concerne le lecteur de disquettes.

FIRMWARE

En anglais, FIRMWARE signifie "MACRO-PROGRAMMATION". Vous trouverez dans un firmware tout ce qui concerne la programmation dite système de votre machine, c'est-à-dire la programmation de l'ordinateur au plus faible niveau, généralement celui du bit.

Au début de l'informatique, les constructeurs avaient plutôt tendance à conserver jalousement leurs secrets de fabrication, ou les spécifications techniques de leurs machines, pour on ne sait quelle obscure terreur du plagiat. Puis, bien vite, certains constructeurs se sont aperçus que plus les informations circulaient à propos de leurs machines, plus elles se vendaient bien, car plus les développeurs s'y attachaient. C'est pourquoi de nos jours, les manuels de références techniques ou firmwares apparaissent généralement quelques semaines après la sortie d'une machine, voire en même temps.
A l'époque de l'annonce du premier PC, IBM sortit le manuel de références techniques, un mois seulement après la sortie de son petit dernier.

Ce que vous trouverez dans un firmware concerne tout d'abord l'organisation de la mémoire, les références techniques de tous les processeurs, et leurs accessibilité (adresse des ports d'entrées sorties en mémoire) ; en règle générale, chacun des bits de ces ports est programmable, et a une signification et une utilisation propre, d'où le terme de macro-programmation du fait que chacun des bits d'un octet puisse avoir une signification.
Un firmware contient également des informations sur les vecteurs de saut dans une ROM d'exploitation ou dans un BIOS. L'erreur à ne pas commettre lorsque l'on achète un firmware est d'attendre de lui qu'il vous explique comment programmer une machine. Ce type d'ouvrage n'est généralement qu'une suite

F

d'informations brutes sur une machine, sans explication détaillée autre que celles nécessaires à la compréhension de son fonctionnement. Le type d'informations contenues dans ce type d'ouvrages s'adresse généralement à des programmeurs de niveau élevé, sachant théoriquement en extraire les informations qu'ils désirent. Si vous désirez apprendre à programmer votre machine en assembleur, par exemple, il est préférable d'avoir à la fois un guide de programmation pour apprendre, et un guide de références techniques (firmware) pour obtenir les informations sur le système qui vous seront nécessaires.

FONCTION

Une fonction est par exemple dans un programme BASIC, un opérande qui va permettre de produire derrière une instruction un résultat. PEEK par exemple est une fonction car on peut la placer derrière une instruction de type :

A = PEEK 23456

Alors que POKE est une instruction qui permet de réaliser une opération comme exemple :

POKE 23456,24.

Une méthode simple pour déterminer si un mot-clé est une fonction ou une instruction, consiste à le placer derrière une instruction connue par exemple PRINT.

Ainsi :

PRINT POKE 23564 est impossible alors que :

PRINT PEEK, 23456 ne pose aucun problème.

FORTH

Les langages de l'informatique sont, paraît-il, classés en plusieurs catégories. La première génération, la seconde génération, et ainsi de suite jusqu'à la cinquième génération. Des langages tels que PROLOG sont dits de 5ème génération, et des langages tels que PASCAL sont de quatrième génération. Ceci étant dit, effectuons maintenant un bref historique du langage FORTH.

. 1970 -
Charles MOORE, pour des applications d'astronomie, écrit un nouveau langage, qui lui semble très évolué. Il est tellement évolué ce langage, que Charles a l'impression qu'il s'agit du premier langage de type quatrième génération (FOURTH signifie "quatrième" en anglais).

. 1970 à 1973 -

Le FORTH se répand au sein de l'astronomie mondiale, en s'envolant d'un observatoire à un autre à travers des modems.

. 1973 -
FORTH inc. est crée par Charles Moore pour promouvoir le FORTH (décidément les Americains ont le sens des affaires !).

. 1973 à 1980 -
Le FORTH se développe et existe sur pratiquement tous les ordinateurs.

. 1979 -
Des groupes d'utilisateurs se créent. Ils se nomment le FORTH Interest Group (FIG qui donnera son nom au FIG FORTH qui est une version du FORTH) aux USA et le FORTH USER GROUP en EUROPE. Ces groupes alliés à FORTH Inc. normalisent FORTH, et créent la norme FORTH 79.

Le FORTH est un langage puissant qui est de type compilé. De nombreux jeux (notamment ceux des cafés) sont développés en FORTH. Sa programmation est de type structurée, et chaque routine créée se transforme en instructions pouvant être utilisées dans un programme. L'exemple significatif qui permet de définir en quelques mots le FORTH est que tout programme développé en FORTH est une nouvelle instruction FORTH qui pourra à son tour être utilisée dans un programme. On obtient ainsi une gigantesque arborescence, base d'un système très performant. FORTH devrait d'ailleurs pouvoir être une excellente base pour créer un système d'exploitation.

Il existe plusieurs dialectes de FORTH, surtout dans le "dompub", autrement dit le domaine public. Le dernier en date (voir Amstrad Magazine n° 22) est un F 83 très performant proposé par l'association Jedi. Il se compose, en plus du langage, d'un assembleur intégré, d'un éditeur, d'un debugger, d'un décompilateur et d'un métacompilateur !

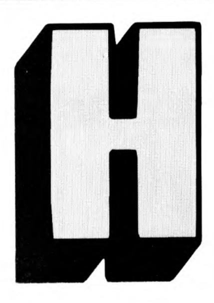
L'association OUF a aussi un FORTH à son catalogue.



GEM

Dans le monde des ordinateurs, il existait un problème qui semblait inconsistant aux programmeurs, et primordial pour les utilisateurs néophytes (cruelle différence de point de vue, reconnaissons-le, en défaveur des seconds). Pour les personnes n'ayant jamais touché un ordinateur de leur vie, une question cruciale se posait sans arrêt : comment toucher un ordinateur sans devenir un virtuose de la programmation ?

Pendant ce temps, une équipe de programmeurs géniaux, dans le plus grand secret des laboratoires d'APPLE à CUPERTINO, s'acharnait pour rendre un ordinateur convivial, et donc à la portée de tous. Au bout de trois longues années de dur labeur, une révolution était née, elle s'appelait MACINTOSH (rendons toutefois à LISA ce qui fut à LISA, le LISA d'APPLE existait bien avant le MAC et en avait toutes les fonctionnalités, mais son prix et sa conception en avaient fait un fiasco).
Face à cette machine géniale, les programmeurs bien-pensants se dirent que le PC d'IBM devenait bien triste, et qu'il fallait, lui aussi, qu'il ait sa souris, sa fenêtre, et ses icônes : GEM était né chez DIGITAL RESEARCH. A dire vrai, le PC sous GEM devenait si semblable au petit dernier d'APPLE que cette firme intenta un procès à DRI, qui le perdit et GEM dut être modifié.
GEM est ce qui est couramment appelé un environnement graphique. Chaque élément est représenté sous la forme d'un symbole graphique : une icône. Ainsi, si vous désirez voir le catalogue d'une disquette, plus besoin de taper DIR au clavier, il suffit de cliquer sur le petit dessin qui représente une disquette et vous verrez apparaître à l'écran les fichiers sous forme de dossiers ou de dessins pour les programmes (par exemple un pinceau pour GEM PAINT qui est un logiciel de dessin). Des fenêtres modulables représentent les zones de travail, et tout texte tapé peut avoir, à l'écran, diverses tailles ou styles de police. Bref GEM ça vous transforme un PC en ordinateur sympa !



HARDCOPY

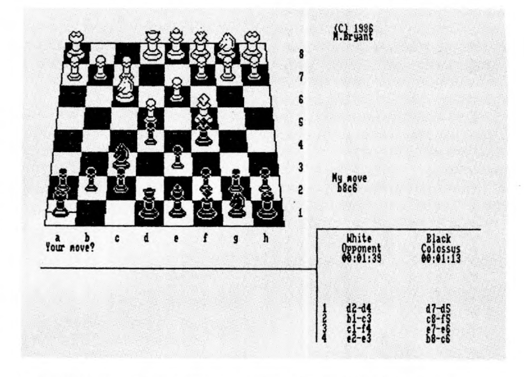
Le hardcopy se présente généralement sous la forme d'une routine disponible à tout instant dans un programme. Une routine de hardcopy est une routine servant à la recopie d'un écran (on dit également vidage), sur une imprimante.

Le rôle de cette routine sera de prendre chaque élément de l'écran et de le reproduire sur une imprimante. Il existe plusieurs sortes de routine de hardcopy : graphique ou ASCII. Les routines de hardcopy graphique peuvent elles-mêmes être divisées en plusieurs catégories, selon qu'elles s'adressent à une imprimante couleur, laser, ou matricielle. Les routines de hardcopy ASCII se contentent elles de saisir chaque caractère, et de le reproduire sur une imprimante avec le code adéquat. Le problème de ce type de routine, est que bien souvent, en raison du manque de standardisation des imprimantes et des écrans (ou cartes graphiques), elles nécessitent une réécriture pour chaque périphérique utilisé. On parle dans ce cas de drivers.

H

GEM et son utilitaire de sortie GEM OUTPUT utilisent ainsi des drivers d'imprimantes. L'utilitaire TASCOPY de la société SEMAPHORE est une routine de hardcopy destinée à la reproduction d'écrans graphiques réalisés sur CPC, sur une imprimante (quel que soit son type). Le programme GRAPHICS des IBM PC et compatibles est un utilitaire de recopie graphique disponible à tout moment, et qui permet de reproduire le contenu de l'écran de l'IBM (ou de l'AMSTRAD PC) sur une imprimante présélectionnée. Le mode de fonctionnement d'une routine de recopie graphique est très simple. Examinons pour commencer une routine de recopie d'écran ASCII. Selon le matériel sur lequel nous travaillons, la routine va saisir directement en mémoire le caractère, l'interpréteur, et le recopier (cas des IBM PC) équipés de carte dite ASCII, ou de l'AMSTRAD PC en mode 0 à 4). L'avantage de ce type de routine est sa rapidité (généralement 80 fois 25 caractères à recopier soit 2000 caractères au maximum). On peut d'ailleurs les "ergonomiser", c'est-à-dire les rendre les plus performantes possibles. Par exemple si une ligne ne comporte que trois caractères en son début, la routine reproduira ces trois caractères sur l'imprimante, puis passera automatiquement à la ligne suivante, sans aller jusqu'au bout de la page, ce qui permettra de gagner beaucoup de temps.

Les routines de recopie d'écrans graphiques ASCII sont un peu plus complexes. En effet, sur l'écran d'un CPC, même en mode texte, les caractères affichés ont une image binaire (c'est-à-dire graphique) en mémoire vidéo. Il faut donc recopier l'écran en interprétant chacun des caractères, mais sans néanmoins utiliser les codes graphiques de l'imprimante qui sont beaucoup trop lents. Nous disposons pour cela en mémoire morte d'une routine qui offre la possibilité d'analyser chacun des caractères très rapidement, puis de les diriger en ASCII sur l'imprimante. Néanmoins, si un pixel graphique encombre l'écran, la routine ne le reconnaitra pas, et un blanc risque d'apparaître sur l'imprimante. Pour les routines de recopie graphique pure, chaque pixel est examiné, puis reproduit sur l'imprimante. si plusieurs couleurs sont affichées, il faut en plus analyser cette couleur pour reproduire différents niveaux de gris sur l'imprimante, ou de couleur sur une imprimante couleur graphique.



HEADER

Un header est une en-tête. Mais une en-tête de quoi me répondrez-vous? Et bien une en-tête de n'importe quoi. En informatique, de nombreux éléments logiciels ont un header. Ce header est toutefois le plus souvent rencontré dans le cas de fichiers sur disquettes ou sur cassettes. Néanmoins, la notion de header signifie zone de données (table de stockage), contenant des informations sur les données qui vont suivre.

Cette zone va contenir une foule d'informations sur l'élément à traiter. Dans le cas d'un fichier par exemple, le header peut contenir des renseignements sur le type du fichier, sa longueur, sa date de création, son nom, sa localisation sur le disque, etc. Il peut y avoir des headers dans bien d'autres cas, par exemple au début d'une ROM pour spécifier les paramètres qui doivent permettre au

système d'exploitation de localiser son point d'entrée et de l'initialiser. Prenons l'exemple d'un header de fichier MS-DOS : celui-ci contient des informations sur la date de création du fichier, et celle de la dernière modification qu'il a subi. A chaque fois que vous allez réaliser un directory de la disquette, le système pourra afficher la date de la dernière modification afin de vous permettre de repérer le fichier plus facilement. Si vous faites un backup de ce fichier, l'attribut de BACKUP situé dans le header va être placé de telle manière que lorsque vous effectuerez une prochaine procédure de BACKUP, le fichier ne soit pas sauvegardé à nouveau s'il n'a pas été remodifié depuis.

Voici à quoi peut servir un header. Sur les PC, vous avez également des headers dans les ROM, qui permettent lors de l'initialisaiton du système de connaître la longueur d'un bloc programme situé dans une ROM spécifique, et de spécifier le point d'entrée du logiciel de cette ROM. Il est parfois possible, avec un système d'exploitation, de connaître le contenu d'un header.

Par exemple sous CP/M, si vous tapez la commande :
DIR [FULL] : toutes les données des headers des fichiers seront affichées à l'écran. Il en est de même sous DOS Plus avec la commande SDIR. Certains logiciels du commerce permettent également de modifier le contenu des headers. Lorsque vous effacez un fichier sur une disquette, par exemple, seule la première lettre du nom de fichier situé dans le header est effacée. Il est possible en remettant cette lettre à son état initial de retrouver le fichier. Pour en terminer avec les headers, nous allons parler de ceux de logiciels sur cassette, qui sont de loin les plus importants. En effet un logiciel sur cassette est bien moins simple à identifier qu'un logiciel sur disquette. De fait, les en-têtes des fichiers sur cassettes sont très pratiques. Ils permettent à une routine de chargement de connaître la longueur du fichier pour dépister toute erreur éventuelle de chargement. Ces même headers sont d'ailleurs parfois non standards et lisibles uniquement avec une routine spéciale, ceci afin de décourager les éventuelles tentatives de copie.

HORLOGE

Tout comme le corps humain qui a besoin des poumons, du cœur et des cellules nerveuses pour exister, l'ordinateur pour fonctionner a besoin d'une horloge, de courant électrique et d'un microprocesseur. Dans cet article nous porterons notre attention essentiellement sur l'horloge.

Mais avant de parler de l'horloge elle-même faisons un peu d'histoire. La piézoélectricité est un phénomène caractérisé par l'apparition de charges électriques de signes opposés (+q et -q) sur deux faces particulières d'une lame, taillé dans un cristal non conducteur, quand une contrainte mécanique, traction ou compression (le vide pour un quartz), est exercée sur une lame suivant une direction donnée. L'effet piézoélectrique a été découvert en 1817 par le minéralogiste français René-Just Hauy : les lois régissant ce phénomène ont été découvertes en 1880 par les frères Jacques et Pierre Curie. L'effet piézoéléctrique est particulièrement sensible avec le quartz. Mis à part que cet effet trouve son application dans la mesure de pressions, dans l'étude des vibrations mécaniques de fréquences subsoniques, soniques ou ultrasoniques, on utilise des lames de quartz pour des horloges de microprocesseurs. Une lame de quartz est un solide élastique qui présente une fréquence propre de vibration mécanique f0 (dilatation et compression) dont la valeur est donnée par la formule : f0 = v/2*e avec v=vitese de propagation dans le quartz (3500 m:s) et l'épaisseur de la lame, épaisseur égale à une demi-longeur d'onde de la vibration. Donc pour une épaisseur de 1mm on trouve f0=3500/2*0.0001=1.75 Mhz. Pour les CPC les quartz sont à 16 Mhz ; on en conclut que l'épaisseur de la pastille de quartz est d'environ 0.2 mm. Attention, ça ne veut pas dire que le microprocesseur fonctionne à 16 Mhz. Sur les CPC, il fonctionne à environ 4 Mhz (quatre millions de vibrations par seconde !!). Lorsqu'une lame de quartz vibre à cette fréquence f0, entre deux électrodes convenablement disposées sur cette lame, on obtient une tension variant à cette même fréquence f0. La lame de quartz est alors comparable à un circuit électrique dit oscillant comportant une inductance L, une résistance R et deux

condensateurs C1 et C2 associés comme indiqué sur la figure 1. Cette lame peut remplacer un circuit oscillant dans un montage électronique *conçu pour qu'une oscillation* (en général provoquée) déclenche un processus d'entretien du régime oscillatoire. L'ensemble fournit un oscillateur pilote délivrant une tension de fréquence bien constante f0. Sur la figure 2 vous voyez un exemple d'horloge employé dans un CPC. Les deux premières NAND, les deux condensateurs C1 et C2 et les deux résistances sont responsables des oscillations provoquées. Le quartz assure une régulation parfaite de la fréquence à f0. Quant aux portes 2 et 3, elles remettent le signal rectangulaire en forme en cas de déformation accidentelle. Le condensateur C3 absorbe les parasites éventuels. Comme on peut le constater il s'agit d'une horloge externe aux circuits intégrés. Il existe dans des systèmes beaucoup plus performants des horloges intégrés. Le principe est le même mais les oscillations sont plus sévèrement contrôlées. En effet pour des machines fonctionnant 24 heures sur 24 il existe un léger glissement de fréquence qui se produit dans l'oscillateur à quartz (voisin de *0.0001 seconde par jour*). Pour éviter ceci, on l'incorpore dans un système asservi qui a pour but de rectifier l'erreur commise. Pour mesurer cette erreur il faut une fréquence référence qui soit stable et parfaitement définie. Cette fréquence étalon est celle de l'onde émise ou absorbée lors du passage d'un électron entre deux niveaux d'énergie déterminés dans un atome. La précision est de l'ordre de 2x10^(-11) seconde par mois. Ces deux types d'horloges sont employés dans presque tous les systèmes car leur précision est élevée. On peut également à partir du 50 hertz du secteur puis à l'aide d'un transformateur, d'un pont de *diodes, d'un trigger de Schmitt* et d'un quartz obtenir une horloge. C'est d'ailleurs ce système qui est utilisé pour les réveils électroniques. Le problème est que la fréquence du secteur est loin d'être régulière, ce qui entraîne souvent un retard de plusieurs secondes par mois. Nous espérons que vous pouvez maintenant saisir la complexité (croissante) qu'abordent les ingénieurs lorsqu'ils ont à concevoir une horloge de microprocesseur.

DEDE-LA-BIDOUILLE

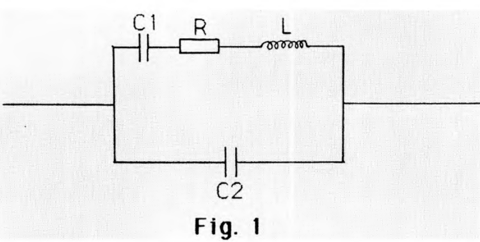


Fig. 1

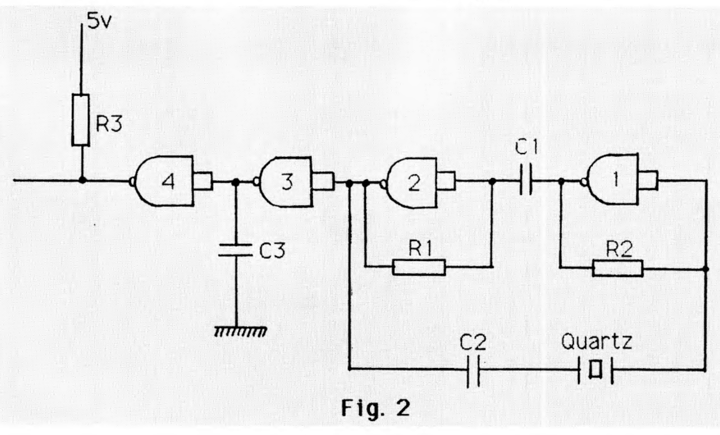
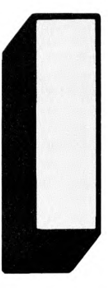


Fig. 2



ICONE

Une icône est un symbole graphique destiné à vous représenter sous la forme la plus explicite possible un élément de l'ordinateur. Ces icônes sont par exemple utilisées sous GEM pour représenter un fichier ou un outil de travail. Par exemple, un fichier de texte sera représenté par une page remplie d'écriture, un programme de dessin par une palette, un curseur pour dessiner sera représenté par un pinceau etc. etc.

L'environnement PC GEM utilise largement cette représentation, mais aussi AMX Page Maker ou les logiciels de la tablette Graphiscop sur CPC.

Vous pouvez modifier ou créer des icônes sous GEM avec l'utilisateur ICON DESIGNER du GEM DEVELOPMENT TOOKLIT vendu en France par Micropol, et conçu par DIGITAL RESEARCH.

INTEL 8086

Le processeur 8086 est comme vous le savez le processeur qui équipe les AMSTRAD de la série PC 1512. Il s'agit d'un processeur 16 Bits de la famille des 8080 d'INTEL, cadencé à 8 Méga-hertz, contrairement aux processeurs de la famille 8088 équipant les IBM PC et compatibles, faux 16 Bits cadencés à 4,77 Méga-hertz.

Le 8086, tout comme le 8088 permet d'adresser jusqu'à 1024 Koctets (1 méga octets) de mémoire. Intel rendit disponible ce microprocesseur en 1978. Il s'agissait du premier 16 bits de cette société, et était 7 à 12 fois plus performant que les 8 bits en technologie MOS. Compatible au niveau du langage source avec le 8080 A et le 8085 A, il offre en plus de nouvelles possibilités d'adressage (voir plus haut) supérieures aux classiques 64 Koctets des 8 bits : le travail au niveau du bit, le traitement et le mouvement de chaînes de caractères, la possibilité d'opérations arithmétiques signées sur 16 bits ainsi que des multiplications et des divisions. Une nouveauté de taille, prémisse de l'architecture dite pipe-line des 80386 (l'architecture pipe-line est une méthode de fonctionnement des processeurs qui permet de traiter partiellement des instructions en file d'attente), permettait avec une file d'attente de 6 octets d'acquérir des informations avant traitement pour accélérer le temps d'exécution. Le 8086 peut travailler sur des mots de 8 ou 16 bits. Il est développé en technologie H MOS et offre sur une plaquette de silicium d'un volume de 32,7 mm2 l'équivalent de 29000 transistors. Sa source d'alimentation unique est de 5 V, et son horloge de base tourne à 5 Mhtz (les versions 8086-2 et 8086-1 permettant respectivement un fonctionnement à 8 et 10 MHtz).

Intel propose autour du 8086 toute une famille de coprocesseurs (dont certains équipent le PC 1512) tels que le 8284 qui est un circuit d'horloge, le 8288 qui est un contrôleur de bus, le contrôleur d'interruption 8259 A, des circuits de puissance pour les bus de données (8286/8287) et les bus d'adresse (8282/8283). Il existe également un processeur d'entrées-sorties dénommé 8089. Si le 8086 fonctionne en mode 8 bits, il est possible de lui adjoindre des processeurs de la famillle 8080 et 8085. N'oublions pas pour clore ce paragraphe sur les coprocesseurs ; le 8087 qui est une unité arithmétique, et qui permet d'accroître notablement la rapidité du 8086 dans les opérations de calcul (un emplacement est prévu sur l'AMSTRAD PC 1512).

Le 8086, nous l'avons vu, peut adresser 1 million d'octets de mémoire (1024 K très exactement). Il dispose pour cela d'un bus d'adresse sur 20 bits, mais pour des raisons de commodité et de compatibilité avec les anciens processeurs, cette mémoire est segmentée en seize pages de 64 Koctets. Le 8086 dispose également de 16 bits pour gérer les entrées-sorties, soit 65535 ports disponibles, contre 255 sur les processeurs 8 bits tels le Z80 ou le 8080. Ce processeur se présente sous la forme d'un boîtier de quarante broches. Il a été conçu pour pouvoir fonctionner dans un environnement multiprocesseurs se

partageant le même bus d'utiliser chacun leurs ressources pour une partie du travail d'un système.

L'organisation interne du 8086 est décomposée comme suit : trois groupes de registre, le premier étant destiné au traitement de données (AX, BX, CX, DX), chacun de ces registres sur 16 bits pouvant être décomposé en deux registres de 8 bits (par exemple AX devient AL et BH). Un groupe de registres d'adressage à l'intérieur d'un segment (SP, BP, SI, DI), et quatre registres de segment (CS, DS, SS et ES).

Les registres de données se comportent comme des registres de stockage ou de travail sur 16 ou 8 bits, et participent tant aux opérations arithmétiques que logiques du 8086. Chacun de ces registres a un rôle particulier qu'il serait trop long de décrire ici. Les registres de segment permettent au processeur d'adresser à tout moment jusqu'à 256 K octets de mémoire. Il existe également plusieurs registres d'état qui permettent d'obtenir des informations après diverses formes d'opération (retenue, signe, etc). Comme vous pouvez le constater, le 8086 est un processeur extrêmement puissant, et il est intéressant de noter que la firme INTEL, quelle que soit la technologie employée cherche toujours à converver une certaine forme de compatibilité entre ses divers modèles. Aujourd'hui, le 8086 est quelque peu dépassé par le 80286 (16/32 bits) qui fut précédé par le 8086, et l'annonce récente des machines à base de 80386 laisse à penser qu'il deviendra aussi obsolète que le 8088 d'ici peu de temps.

INTELLIGENCE ARTIFICIELLE

Depuis bientôt trente ans de maturation dans les laboratoires, le domaine de l'intelligence artificielle (I.A) a cessé de faire partie de la catégorie des rêves fous de l'homme. En effet l'intelligence artificielle est bel et bien implantée dans beaucoup de domaines. Elle tend d'ailleurs à s'intéresser à tous les sujets autant abstraits que concrets. Ainsi l'aide à la décision de divulgations de secrets d'état est un logiciel employé par l'armée. De même les centrales nucléaires utilisent un robot commandé par microprocesseur utilisant l'I.A pour prévenir et réparer des défaillances pouvant surgir au cœur de la centrale.

Mais avant de parler de l'I.A., tentons de bien redéfinir ce qu'est l'intelligence. Pour la psychologie, l'intelligence est la faculté de donner une solution efficace à un problème nouveau d'ordre quelconque. Il est difficile de proposer une définition de l'intelligence indépendante des théories psychologiques qui prétendent la mesurer. On peut cependant dégager les traits suivants :

- l'intelligence est une aptitude variable selon les individus considérés ; elle a des degrés ;
- cette aptitude consiste à savoir donner aux problèmes de tout ordre une réponse originale, adaptée.
- Les solutions intelligentes se distinguent radicalement des solutions obtenues par tâtonnements aveugles (qui ne donnent lieu à aucun progrès), par instinct ou par habitude (solutions inutilisables si des nouveaux problèmes sont posées au sujet.).

Pour les machines, nous ne considérerons que les deux derniers points. Il faut également ajouter qu'il existe plusieurs formes d'intelligence. Nous n'en citerons que trois : l'ntelligence pratique qui (observée chez les enfants) fait entrer des mouvements appropriés pour trouver la solution de petits problèmes concrets (lego, puzzle, boîtes à empiler ...). L'intelligence technique consiste à résoudre un problème par la fabrication d'un instrument ou d'un outil approprié. L'intelligence conceptuelle, verbale, logique résoud les problèmes par l'intermédiaire de signes ou symboles (mots du langage, concepts, plans, opérations intellectuelles de toute espèce). Cette forme spécifiquement humaine d'intelligence intervient, bien entendu dès les niveaux précédents. La fabrication d'un outil, par exemple, est inséparable du langage dans la culture humaine. Après avoir bien assimilé ces concepts de l'intelligence avec les exemples ci-dessous d'aplications des ordinateurs sur différents domaines, il ne devrait plus tellement rester (du moins nous l'espérons) de personnes sceptiques sur

l'existence de l'intelligence (artificielle) sur ordinateurs.
Prenons une expérience très marquante de ces dernières années (1950) réalisée sur deux jeux d'échecs ; l'un tenu par un joueur humain, l'autre par une machine. Un voile cachait les deux joueurs et un adversaire jouait contre les deux et essayait de deviner quel était le jeu tenu par la machine ou l'homme. Le logiciel était bon, il n'a donc pas été possible de séparer (avec un pourcentage décent d'estimation) l'un ou l'autre. L'I.A. prenant toute sa dimension que nous lui connaissons aujourd'hui. Il existe actuellement un jeu d'echec classé mondialement à la dizième place.
Par différents procédés on arrive maintenant à toucher les trois formes d'intelligence citées plus haut. Comment faire raisonner (chose abstraite) un ordinateur (objet concret) ? Quand nous cherchons une solution à un problème nous le définissons puis nos cellules nerveuses agissent spacialement (trois dimensions) pour arriver le plus vite possible à la solution et pour cela on utilise une partie de notre mémoire. Mais on apprend que notre mémoire est associée à des structures situées sur la surface interne des lobes temporaux, comme l'hippocampe mais cette dernière est constituée de cellules interagissant chimiquement entre elles par des phénomènes extrêmement compliqués qui sont loin d'être tous élucidés à l'heure actuelle.
Il a donc fallu inventer un processus ou un objet remplaçant cette mémoire. C'est avec l'intégration de milliers, voire de millions de transistors sur une puce de silicium de quelques millimètres carré qu'on a réussi à remplacer cette mémoire. Maintenant que la mémoire est disponible encore faut-il savoir l'utiliser rationnellement.
C'est ce que va s'efforcer de faire, à l'aide de différents langages adaptés tels que le PROLOG, le LISP, ADA pour ne citer qu'eux, un ordinateur programme dans un but précis.
Prenons un exemple de tous les jours. Vous êtes au volant de votre voiture et vous voyez le feu passer à l'orange. (Situons l'exemple il y a quelques années car maintenant même au feu orange on est invité à verser une somme substantielle non négligeable à ces messieurs responsables de l'ordre public). Il y a une multitude de réactions en fonction des gens (mis à part leur forme physique et leurs réflexes). L'un passera car le feu est orange, donc il accélèrera. Un deuxième s'arrêtera sachant pertinemment que sa voiture non puissante ne lui permettrait pas de passer le carrefour à temps. Un troisième s'arrêtera car le feu est rapide et le risque est trop grand malgré la puissance de sa voiture. Un quatrième avec une voiture puissante, n'aimant pas prendre de risque passera quand même car un camoin est le premier de la file du feu qui va passer au vert, plus le temps que le camoin démarre, il a largement le temps de passer. Imaginez un peu tous les facteurs qui peuvent entrer en plus : par exemple la route est mouillée, la probabilité qu'il y ait une voiture en décélération au moment où le feu passe vert etc ... Vous voyez qu'il est impossible de considérer tous ces paramètres lors de la conduite d'un véhicule. D'ailleurs même si certains sont envisagés, ils sont éliminés d'office car leur probabilité pour provoquer un accident est faible. Mais il faut cependant garder les plus représentatifs, c'est-à-dire estimés pour provoquer à plus de 50 % un accident.
Cet exemple peut paraître simpliste mais c'est exactement la façon qu'utilise un ordinateur pour prendre des "décisions" ; le tout étant de bien le programmer. Et une fois de plus, le concepteur d'un pilote automatique ne considérera pas tous les cas particuliers mais un ensemble de cas généraux. Voilà l'idée force de l'I.A. Des capteurs disposés un peut partout sur le véhicule renseigneront l'ordinateur central géré par un moteur d'inférence (cœur du programme) qui en fonction des différentes données (calculs de probabilités, degré d'importance des paramètres) donnera une solution adéquate au problème posé. Il existe actuellement des programmes en I.A. encore appelés systèmes experts qui sont capables de démontrer des théorèmes de mathématiques de façon beaucoup plus directe que des démonstrations humaines. Certains même ont été démontrés alors qu'ils ne l'avaient jamais été auparavant. Voici de quoi encourager le développement de ces systèmes.
Pour bien fixer les idées faisons un parallèle entre la langue française et le chinois ou le japonais. Ces deux langues orientales beaucoup moins précises que le français entraînent, par des signes appelés idéogrammes, une vision spaciale du sujet traité. Ce sont en quelque sorte des langues déductives. Pour indiquer qu'un homme est fort on dira qu'il a la stature d'un ours. Pour la souplesse on prendra un félin etc... Ce sont des associations. En deux mots on opère une récursivité au niveau de la compréhension. C'est donc par des concepts que l'on arrive plus facilement à englober le maximum de cas et non en définissant chaque objet. La formidable puissance de l'I.A c'est qu'elle peut traiter des problèmes sous forme de concepts (trois dimensions de raisonnement dues à la récursivité) et ne s'arrête pas qu'à

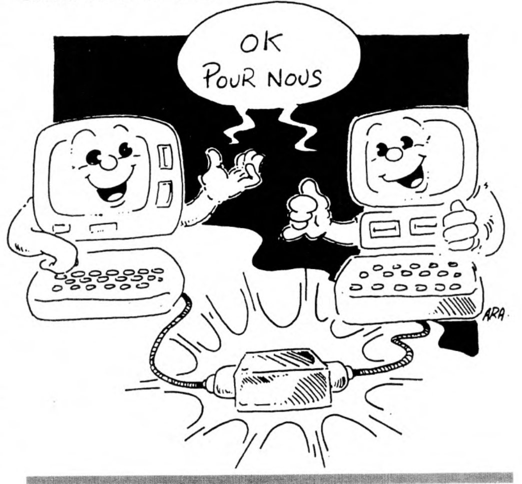
l'analyse pure et simple de *paramètres (deux dimensions* au niveau du raisonnement : vrai ou faux).
Par exemple si vous êtes allé à l'exposition d'AMSTRAD vous avez vu sur un PC 1512 équipé du TURBO-PROLOG la résolution de sept tours de Hanoï en moins d'une minute. Le programme ne fait que sept lignes !!! (et encore, il y a des instructions pour la gestion de l'affichage). Il paraît que sur la résolution de douze tours le PC 1512 équipé du TURBO-PROLOG met plus d'une trentaine de secondes en moins qu'un gros ordinateur VAX équipé d'un simple PROLOG. En tout cas, pour sept tours il existe déjà plus d'un million de combinaisons possibles. Imaginez-vous (même si l'ordinateur travaille vite) s'il fallait envisager toutes les solutions à chaque mouvement !
Le programme est donc là pour optimiser le nombre de coups en "réfléchissant" à celui qui serait le plus avantageux pour la suite. Attention d'anciens programmes d'I.A donnaient le meilleur coup en fonction de l'analyse des coups suivants. Avec par exemple, pour le jeu de dames, une prévision de dix coups à l'avance, l'explosion combinatoire rend vite infernale la résolution du problème. Ce genre de programme est en plus très gourmand en mémoire. L'I.A permet donc d'avoir en quelque sorte une "vision globale" du problème. Plutôt que de partir d'une arborescence multiple, de sauter certain fait jugé inintéressant, de prendre en compte ceux qui sont susceptibles de servir, on définit le problème globalement pour essayer de créer un concept.
Certains systèmes experts à leur naissance ne pouvaient être considérés comme de véritables outils de l'I.A car ils ne déduisaient pas mais donnaient une solution correspondant à des faits précis. C'était plutôt une réponse que l'on peut classer dans les réflexes. A un stimulus extérieur, par exemple une brûlure à la main, on retire sa main de la flamme et on crie. Mais le programme était incapable de donner les autres raisons pour lesquelles on crie.
C'est maintenant possible grâce à la création de ces nouveaux langages spécialisés dans l'I.A. Il y aurait encore beaucoup de choses à dire sur le sujet mais nous serions obligés de rentrer dans des domaines techniques trop spécialisés. Pour approfondir vos connaissances nous vous conseillons le n° 170 d'octobre 1985 de la revue "LA RECHERCHE" (numéro hors série sur l'I.A) ainsi que le livre "Trois étapes vers l'intelligence artificielle" de René Descamps édité chez PSI.
Voici pour finir deux exemples d'application qui donnent à réfléchir : un *hélicoptère de l'armée piloté par* ordinateur détecte une ligne de quatre chars. Il en élimine un dans ses données car ce n'était qu'un rocher avec l'aspect d'un char. Or les trois chars avancent et l'hélicoptère risque de se faire repérer ; il se camoufle donc derrière une rangée d'arbres. Le but serait de détruire les chars un à un. Or ils se déplacent sur une route qui passe sur un pont. Donc le but final sera de sortir de sa cachette, de tirer sur le pont quand les trois chars seront dessus afin de les immobiliser et ceci avec un minimum de risques. Il existe, encore à titre expérimental, des analyseurs de syntaxes capables de traduire instantanément une langue dans une autre. Peut-être avec cette application arriverons-nous à comprendre davantage les autres peuples ? Alors l'I.A. pour nuire ou pour construire ? Et construire dans quel sens ?

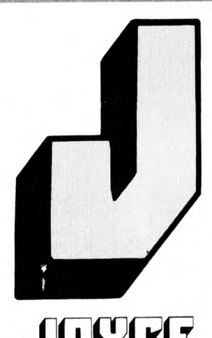
INTERRUPTION

Les interruptions sont des instructions ou des modes de fonctionnement de processeurs qui permettent d'interrompre le travail d'un logiciel selon différents niveaux de priorité ou types d'événements.

En règle générale, une interruption est presque toujours provoquée par le processeur. Il existe les interruptions masquables et les interruptions non masquables. Ces premières peuvent être supplantées par certaines autres sources, les non masquables agiront quoiqu'il arrive. On observe également d'autres types d'interruptions sur les micro-processeurs actuels. Celles-ci correspondent à des instructions, qui, placées dans un programme, donneront la main à une routine située quelque part dans la mémoire, le logiciel reprenant son cours normal après exécution de cette routine. Les instructions INT des processeurs de la famille des 8088/8086 sont de ce type. Ces instructions sont si pratiques quelles ont été utilisées exclusivement comme points d'accès aux routines du BIOS de l'IBM. Ce sont ces mêmes procédures d'interruptions qu'emploie MS-DOS pour toutes ses fonctions. On peut également les utiliser dans un logiciel pour effectuer un travail

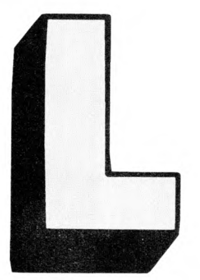
répété (gestion du déroulement d'une horloge par exemple). Il existe aussi des interruptions générées par des processeurs. Prenons l'exemple d'une interface qui, en repos, ne provoque aucune action sur le système, mais qui dès qu'elle va recevoir une information, doit communiquer celle-ci au système. Elle va alors générer une interruption, qui correspondra à une adresse de programme dans la mémoire. Une fois ce programme exécuté (généralement un programme de traitement de données transmises par l'interface), la main sera redonnée au travail en cours avant interruption. La gestion d'un clavier d'AMSTRAD PC se fait par interruption. A chaque fois que vous frappez une touche, le token correspondant est placé dans un buffer , qui sera ensuite lu par la routine de traitement correspondante. Comme vous le voyez, l'interruption est à l'ordinateur, ce que la roue est à la charrue. D'autres types d'interruptions permettent de faire fonctionner plusieurs logiciels en même temps. C'est-à-dire que plusieurs programmes résident ensemble en mémoire, et que le processeur exécute les instructions de chacun de ces logiciels séquentiellement en boucle. Prenons l'exemple de trois logiciels A, B et C. S'ils fonctionnent en multitâches, le processeur prendra l'instruction 1 de A, puis passera à l'instruction 1 de B puis à l'instruction 1 de C. Une fois ce traitement terminé, le processeur reprendra à l'instruction 2 de A, puis l'instruction 2 de B et ainsi de suite, en tenant compte de chaque logiciel. Les logiciels d'arrière-plan de DOS Plus fonctionnent également par interruption (tout comme l'application PRINT de MS-DOS), mais d'une façon légèrement différente. Les trois programmes d'arrière-plan sont exécutés intégralement les uns après les autres, et ainsi de suite en bouclant, ce qui nécessite une certaine gymnastique de programmation. Dernier exemple de routine par interruption, que les plus intéréssés s'y reportent absolument, il s'agit du BASIC LOCOMOTIVE de la famille des CPC 464 qui traite des sous-routines séquentiellement, et qui est extrêmement efficace. C'est d'ailleurs la première et seule fois qu'un tel procédé était implanté dans le BASIC d'un micro familial.





JOYCE

Nom de code donné au PCW 8256 pendant son développement. En fait JOYCE a une double signification : excepté le fait qu'il servit de nom de code, il s'agit également du nom de la secrétaire d'Alan SUGAR, qui s'appelle Joyce KALLEY. Lorsque les études de marketing de la société AMSTRAD démontrèrent qu'un marché très important existait à court terme pour un ordinateur bon marché spécialisé dans le traitement de texte, Alan SUGAR se dit qu'il pourrait ainsi offrir à sa secrétaire le moyen de travailler deux fois plus, et ce pour le même salaire et fit donc baptiser sa machine JOYCE.



LECTEURS DE DISQUETTES

Les lecteurs de la gamme AMSTRAD sont des périphériques très utiles mais aussi très fiables. Si vous avez lu l'article sur les disquettes, vous connaissez alors les avantages de celles-ci sur les cassettes, sinon allez vite le lire. Nous allons ici essayer de découvrir ce périphérique peu connu intérieurement.

Vous pouvez aujourd'hui connecter des lecteurs 3 pouces, 3 pouces et demi ou 5 pouces et quart sur votre CPC grâce à certains constructeurs autres qu'AMSTRAD. Nous ne parlerons ici que du lecteur 3 pouces qui est d'origine avec les CPC 664 et CPC 6128.

Un lecteur de disquette complet est constitué d'une carte contrôleur et d'un bloc semi-mécanique, semi-électronique. Un lecteur sans carte contrôleur peut être utilisé comme lecteur esclave (ou lecteur B). Le composant principal de la carte contrôleur est le FDC 765 (FDC = Floppy Disk Controller = contrôleur de disques souples qui en réalité dirige toutes les informations circulant entre le Z80 du CPC et le lecteur de disquettes. On peut considérer ce circuit comme un micro-processeur très spécialisé ce qui n'est pas exagéré étant donné ses possibilités. C'est un circuit de quarante broches qui fournit tous les signaux nécessaires pour exploiter les lecteurs du marché dans toutes les tailles (3 pouces à 8 pouces). Voici un aperçu des données techniques du FDC :

- longueur de secteur programmable ;
- nombre de secteurs par piste programmable ;
- connectable à presque tous les types de microprocesseurs courants ;
- toutes les données du lecteur programmables ;
- peut gérer des lecteurs de disquettes de toutes tailles simple ou double têtes.

Toutes les possibilités de ce circuit n'ont pas été utilisées par les développeurs de la carte contrôleur. Grâce à la haute intégration du FDC et du séparateur de données, la carte contrôleur ne comporte que très peu de composants électroniques : une quarantaine de petits composants (transistors, diodes, résistances, condensateurs) et une dizaine de circuits intégrés dont le FDC et la ROM contenant le système d'exploitation du lecteur de disquettes.

C'est une ROM de 16 Ko qui contient dans la moitié de sa capacité toutes les routines nécessaires à la gestion du lecteur de disquettes ainsi que certaines routines concernant le lecteur de cassettes. Dans les 8 Ko restants la ROM contient une partie de l'interpréteur LOGO (le LOGO est fourni sur la disquette CP/M 2.2).

Revenons au FDC ; il est situé sur les adresses de port &FB7E et &FB7F où se trouvent respectivement le registre d'état principal et le registre de données. Il y a cependant une troisième adresse qui est &FA7E où se trouve un circuit bascule qui commande les moteurs des lecteurs de disquettes connectés au CPC. Essayer par exemple de taper en Basic un "OUT &FA7E,1", les moteurs de tous les lecteurs seront alors mis en marche. Il vous faudra taper "OUT &FA7E,0" pour les arrêter.

Sauvegarde séquentielle des données

L'utilité d'un lecteur de disquettes n'est pas simplement de permettre la sauvegarde ou le chargement de programmes, il permet aussi la sauvegarde d'un grand nombre de données ce qui était impossible à maîtriser correctement avec un lecteur de cassettes. Les lecteurs de disquettes des CPC ne permettent en principe que de traiter des fichiers séquentiels, c'est-à-dire caractère par caractère. Pour lire le dernier caractère d'un fichier, il faut lire auparavant tous les caractères précédents. Le principe est donc le même qu'avec un lecteur de cassettes mais la vitesse de lecture étant très rapide, on ne s'en rend pas compte.

L'accès direct permet lui, d'accéder directement au caractère voulu et donc plus rapidement. Vous pouvez demander pourquoi les développeurs du système d'exploitation du lecteur de disquettes ont choisi un accès

séquentiel plutôt qu'un accès direct ? C'est tout simplement parce que c'est la méthode la plus simple et donc celle qui est la plus courte à programmer. De plus, étant donné la différence de vitesse d'accès entre les deux méthodes, l'accès séquentiel était plus rentable.

Quelques adresses utiles

Grâce à quelques POKE judicieux, vous pourrez modifier à votre guise certains paramètres du lecteur de disquettes. Nous vous conseillons toutefois de noter les valeurs d'origine qui peuvent vous servir de référence.

- &BE44, &BE45 : délai d'attente après MOTOR ON ;
- &BE46, &BE47 : temps de fin de rotation du moteur du lecteur de disquettes après le dernier accès ;
- &BE66 : nombre de tentatives de lecture ;
- &A8A3 : octet de remplissage pour le formatage d'une piste.

Grâce aux valeurs que vous mettrez en &BE44, &BE45, &BE46, &BE47, vous pourrez accélérer ou ralentir vos opérations sur disquettes. Prenez garde à la valeur que vous mettrez en &BE66, si votre disquette est usagée, il est alors souhaitable de faire plusieurs tentatives de lecture car une disquette usagée est moins fiable qu'une disquette neuve.
La valeur de l'octet que vous mettrez en &A8A3 est un gadget qui n'a pas d'utilité réelle pour vous mais qui peut vous permettre de personnaliser vos disquettes.
Si vous souhaitez plus de renseignements sur le lecteur de disquettes de l'AMSTRAD, nous vous conseillons le livre du lecteur de disquettes qui concerne les trois modèles de CPC.

Eric MISTELET

LOGICIELS

Qu'est-ce donc qu'un logiciel? En fait sous cette appellation, se regroupent plusieurs catégories de choses. Un logiciel est une suite d'instructions dans un quelconque langage informatique, permettant de faire vivre un ordinateur, et de lui faire exécuter des tâches précises. Il existe de nombreuses catégories de logiciels, pouvant chacune regrouper une application très précise. La définition exacte de logiciel pourrait être un nom très vague regroupant toutes les catégories d'applications micro-informatiques. Une machine sans logiciel est une machine morte, elle n'a pas d'intelligence, elle est réduite à sa plus simple expression, c'est-à-dire un tas de composants électroniques sans fonction précise. On peut distinguer dans la famille des logiciels plusieurs noms :

- Progiciel : logiciel à vocation professionelle; il peut s'agir d'une comptabilité, d'un système de paye, de facturation, etc. etc.
- Programme : il s'agit ici en fait d'un synonyme de logiciel.
- Application est également un synonyme de logiciel.

On peut également distinguer d'autres types de logiciels très spécifiques : un langage est un logiciel, tout comme un système d'exploitation tel MS-DOS ou CP/M. Le BIOS d'un PC compatible, contenu en ROM est un logiciel. GSX ou GEM sont des environnements graphiques, mais s'apparentent également à des logiciels.



MEMOIRES

L'architecture d'un ordinateur est à peu près constituée des éléments suivants : unités d'entrée/mémoire principale/unité de traitement/unité de sortie. M comme mémoire : les mémoires sont des unités où peuvent être mises en réserve les données, les instructions et les résultats partiels. Les informations sont fournies sous forme d'impulsions

électriques adaptées aux éléments qui composent ces unités de mémoires. La représentation conventionnelle choisie dépend du code adopté et des principes mis en oeuvre pour conserver l'information.

Une mémoire est divisée en sections ; chacune d'elles peut recevoir une information et est repérée à l'aide d'un nombre qui constitue son adresse. Celle-ci permet à l'opérateur de savoir à quel endroit de la mémoire se trouve l'information. Lorsqu'une information est placée en mémoire, tout enregistrement antérieur est automatiquement effacé ; toutefois, la lecture de la mémoire n'entraîne pas la destruction de l'information mémorisée. Une mémoire est caractérisée par sa capacité, c'est-à-dire par la quantité d'informations qu'elle peut contenir. Dans certains cas, il est possible d'introduire plusieurs centaines de millions de chiffres binaires. Le temps nécessaire à l'obtention d'une information placée en mémoire, appelée temps d'accès, doit être le plus petit possible car, comme les transferts d'informations entre la mémoire principale et l'unité centrale sont très nombreux, de ce temps d'accès dépend l'efficacité de l'ordinateur. Avant de considérer directement la conception de l'AMSTRAD, il est intéressant de connaître un peu les différents types de mémoires existants sur le marché.

. Les cartes perforées peuvent être considérées comme des mémoires externes, car elles ne font pas partie de la machine. Une information est représentée par un groupement de trous disposés suivant un code déterminé. Ces mémoires ont une capacité très grande mais leur temps d'accès dépasse la seconde ce qui est assez long.

. Les mémoires magnétiques : composées d'un matériau ferromagnétique, elles représentent le chiffre 1 pour une aimantation déterminée et le chiffre 0 dans tous les autres cas.

Les mémoires magnétiques statiques se composent d'un ensemble de tores en ferrites, de petite dimension, disposés à l'intersection des deux conducteurs perpendiculaires qui servent à définir la position du tore dans la mémoire (son adresse), à l'aimanter (enregistrement), et également à déterminer l'état dans lequel se trouve la mémoire (lecture). Une mémoire de grande capacité doit comporter un grand nombre de tores car chacun d'eux ne permet la mise en réserve que d'une seule unité d'information. Ces mémoires prennent de la place mais leur temps d'accès est faible (1 à 3 microsecondes).

. Les mémoires magnétiques mobiles comportent un matériau ferromagnétique déposé sur un support (bande, disque, etc.) à la surface duquel il forme une couche extrêmement mince. L'enregistrement et la lecture se font de la même manière que dans un magnétophone, et dans ces derniers dispositifs une zone aimantée figure un signe binaire. Les bandes magnétiques sont des mémoires de grandes capacités et de temps d'accès assez long (10s).

Les mémoires ferro-électriques mettent en application un principe très proche de celui qu'on emploie dans les mémoires magnétiques ; il consiste à utiliser le phénomème d'hytérésis éléctrique présenté par certains isolants. La polarisation d'une portion de diélectrique dépend du sens du champ électrique dans lequel on place l'isolant ; on fait correspondre à un état polarisé le chiffre 1, et le chiffre 0 se trouve défini pour tout état différent du précédent. Grande capacité et temps d'accès assez faible (1 microsec) sont deux avantages mais un défaut important limite leur utilisation : elles ne conservent pas l'information très longtemps, au maximum quelques jours.

. Les mémoires holographiques sont constituées d'un cristal ferro-électrique en niobate de lithium. Les informations sont enregistrées par un laser à argon dans les plans de mémoire dont la position dépend de l'incidence du faisceau laser. La lecture s'effectue par photodiodes, et l'effacement par chauffage local au moyen d'électrodes déposées sur un support de silice. La capacité de ces mémoires est de l'ordre de 10 puissance 12 bits avec un temps d'accès d'environ 1 nanoseconde.

. La mémoire à bulle est constituée de microcanaux de 0.1 micromètre de diamètre. Leur capacité peut atteindre 10 puissance 18 bits par cm carré.

. Enfin celles qui nous concernent : les mémoires à semi-conducteurs. Elles ont un temps d'accès 10 à 100 fois plus faible que celui des tores de ferrite et un encombrement réduit. Une pastille de 25 mm carré est capable de contenir plus de 10 000 transistors ! Il en existe différents types :

. Les ram (random access memory) sont des mémoires dites volatiles car lorsqu'elles sont mises hors tension toutes leurs données s'effacent. Elles peuvent être utilisées en lecture et en écriture. Il y en a deux types : les ram statiques. Leurs cellules élémentaires sont des LATCH (voir fig 1). Leur structure est relativement compliquée, coûteuse et encombrante ce qui réduit leur capacité par boîtier. Les autres sont des ram dynamiques (dans l'Amstrad). La cellule mémoire est la capa-

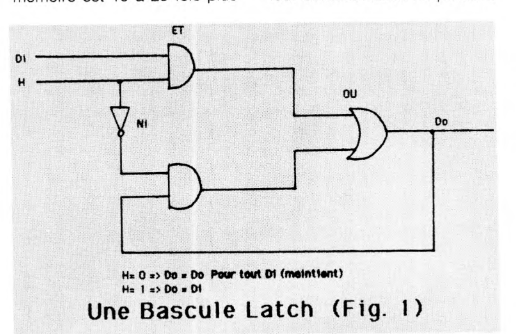
ctié grille-source d'un transistor MOS mais l'information fuit d'où la nécessité de la régénérer par des circuits relativement complexes de rafraîchissement. La capacité mémoire est 10 à 20 fois plus grande que pour une ram statique du fait de la simplicité de la cellule élémentaire mais les contraintes de refresh ne rendent ce type de mémoire compétitif que pour des capacités assez importantes.

.Les rom (read only memory). Elles sont aussi appetées mémoires mortes car lorsque vous coupez l'alimentation elle conservent quand même leur contenu. Elles ne peuvent être utilisées uniquement qu'en lecture (voir fig. 2). Au début les diodes sont partout présentes. La destruction sélective des diodes, réalisée par le fabricant, permet la programmation définitive. Le coût est relativement faible pour une capacité importante.

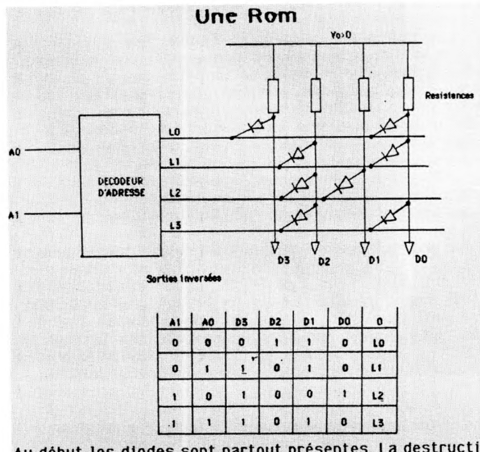
. Les Prom (programmable rom). C'est l'utilisateur qui vaporise les fusibles.

. Les eprom (erasable prom). La cellule mémoire élémentaire est un transistor MOS à grille isolée (par la silice). (voir fig 3). On les efface par ultra-violet.

. Eprom, eaprom (electrically erasable prom) sont des prom effaçables électriquement. Leur durée de vie est donc considérablement augmentée.



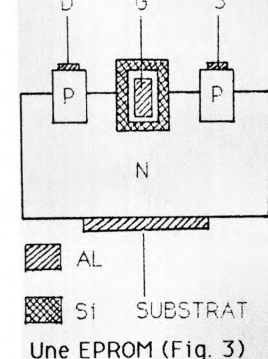
Une Bascule Latch (Fig. 1)



A1	A0	D3	D2	D1	D0	0
0	0	0	1	1	0	L0
0	1	1	0	1	0	L1
1	0	1	0	0	1	L2
.1	1	1	0	1	0	L3

Au début les diodes sont partout présentes. La destruction sélective des diodes permet la programmation définitive.

(Fig. 2)



Une EPROM (Fig. 3)

particularités et place des différentes mémoires pour une bonne utilisation de l'espace mémoire de l'AMSTRAD CPC 464

Donc l'AMSTRAD possède deux types de mémoire : les rams et les roms. Leur place est une particularité due au génie des concepteurs de ce système : elles sont superposées en haut et en bas de l'espace disponible en l'oc-

currence 64 Ko allant de #000 à #ffff. L'ordinateur ne faisant qu'une chose à la fois on aura ou la rom ou la ram de sélectionnée. Ce sont des routines appelées restarts qui s'occupent de la commutation de l'une à l'autre. Voici comment sont utilisés les quatre blocs de 16 k (voir fig. 4).

Pour le 1er bloc en ram :

Les huit restarts sont situés de #0000 à #0040 ; de #0041 à #0170 on a une zone de données utilisables par le système uniquement, puis la zone de #0171 à #4000 est réservée à l'utilisateur.

En rom de #0000 à #4000 ce sont les mêmes restarts qu'en ram puis de #0171 à #4000 il y a le système d'exploitation. C'est en quelque sorte le coeur de l'ordinateur.

Pour le 2ème bloc on a uniquement de la ram utilisable pour des programmes.

Le 3ème bloc est aussi seulement en ram et est divisé comme suit : de #8000 à =ab79 utilisé pour les programmes ; de #ab80 à #b900 utilisé comme zone de données pour le système ; de #b901 à #c000 ce sont des routines que peut appeler l'utilisateur.

Enfin le dernier bloc (ram) où se trouve la mémoire écran et en rom l'interpréteur Basic. C'est cette partie de la mémoire qui reconnaît les mots Basic que vous utilisez pour vos programmes. A ce niveau on peut ajouter 16 Ko. La rom du lecteur de disquettes est d'ailleurs placée ici. De tout ceci, on peut déduire un certain nombre de choses : la zone pour programmer est de #ab79 moins #0171 ce qui fait environ 42 Ko de libre. Le système réserve deux espaces pour des données. L'espace total est d'environ 4 ko ce qui est beaucoup, mais quand on pense au dimensionnement d'un tableau ou la multitude de boucles de certains programmes, on est bien content de les avoir.

Guillaume PONTICELLI

MS-DOS

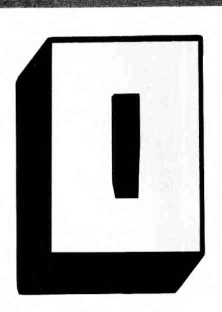
MS-DOS est un système d'exploitation commercialisé par la société MICROSOFT. Ce système dans sa version 3.2 équipe les nouveaux micro-ordinateurs AMSTRAD de la série PC 1512. Il est destiné aux micro-ordinateurs fonctionnant à base de micro-processeurs de la famille du 8088 d'Intel (8088, 8086, 80186, 80286, 80386) et non exclusivement les compatibles IBM PC comme certains semblent le croire. Effectuons un bref historique du système d'exploitation MS-DOS.

En 1980, la firme Seattle Computer Products crée un ordinateur à partir de l'ultime micro-processeur 16 bits d'intel, le 8086. Cet ordinateur s'intitule le S-100 et ne possède aucun système d'exploitation. Pour combler cette lacune, un ingénieur logiciel du nom de Tim Patterson crée le Q-DOS (QUICK and DIRTY Operating System) qui signifie en français : Système d'exploitation Rapide et Sale). Rapide car ce système est réellement très rapide, et sale, car il n'est absolument pas convivial. A la fin de l'année 1980, le système QDOS a considérablement évolué, et change de nom pour s'intituler 86-DOS (DOS du microprocesseur 8086. Pressentant la sortie prochaine d'un micro-ordinateur de la société IBM fonctionnant à base de 8086 (on sait maintenant que le processeur qui équipa le PC ne fut pas le 8086 mais le 8088), Bill GATES, chairman de la société MICROSOFT fait acheter les droits de 86-DOS. IBM sentant le marché suffisamment mûr pour son PC décida de lancer dans le plus grand secret un appel d'offre pour trouver un système d'exploitation capable d'équiper sa machine (c'était la première fois qu'IBM faisait appel à des sociétés extérieures pour développer l'un de ses logiciels, et tout le monde sait à quel point ils s'en mordent les doigts aujourd'hui). MICROSOFT se mit donc au travail, et TIM PATTERSON rejoignit l'équipe en Mai 1981. MICROSOFT racheta à Seattle Computer l'exclusivité des droits de 86-DOS. Le PC d'IBM sortit alors équipé d'un DOS écrit pour lui par MICROSOFT. Ce qui était déjà MS-DOS s'appelait pour la circonstance PC-DOS, par un choix délibéré de la société MICROSOFT. MS-DOS fut très proche de CP/M, pour permettre aux utilisateurs de machines sous CP/M, ainsi qu'aux développeurs d'effectuer une transition facile. L'idée était géniale et le résultat heureux. Dès sa sortie sur le marché, le PC d'IBM était déjà équipé de nombreux logiciels, MS-DOS, mais s'il copiait honteusement CP/M (ce qui valut d'ailleurs à la société DIGITAL de perdre le marché avec sa version 8086 du CP/M, CP/M86) il n'en était pas moins remarquablement conçu. La gestion des disques était bien plus évoluée, avec la possibilité de créer des répertoires et des sous-répertoires, bien plus pratiques pour classer les dossiers et les fichiers que les USER de CP/M. Ce système est également équipé en standard de plusieurs utili-

taires très importants : EDLIN qui est un éditeur de texte, et DEBUG qui est un utilitaire de mise au point et de debuggage. Ensuite, apparut le Macro Assembleur qui, même aujourd'hui confirme son efficacité. Insatiable, MASM accompagné d'un éditeur de liens le LINK MICROSOFT commercialisa une quantité faramineuse de langages pour son système, qui de nos jours demeurent des références indiscutables. MSCOBOL, MSBASIC, BASCOM (compilateur basic), MSPASCAL, MSC. Il y en a pour tous les développeurs, ce qui fait sûrement le succès incontestable de cette société ainsi que celui de son système d'exploitation.

MS-DOS est un système d'exploitation à la fois simple d'utilisation, évolutif et très puissant. Il offre la possibilité à son utilisateur de configurer totalement son système avec une option pour redéfinir le prompt (le petit signe A> qui s'affiche après chaque commande). Il gère l'heure aussi bien en provenance du BIOS des PC que d'une horloge temps réel avec des commandes telles que DATE et TIME. Des fichiers de commandes dits fichiers BATCH permettent de construire de véritables petits programmes utilisant les commandes du système, et un fichier spécial intitulé AUTOEXEC.BAT, qui est également un fichier de commandes exécutables dès que l'on réinitialise le système. MS-DOS peut gérer la copie de fichiers avec l'instruction COPY, des interfaces séries avec l'instruction MODE, et peut même déporter sa console de visualisation et de saisie avec CTTY (c'est-à-dire que tout le contrôle du système peut être renvoyé à un terminal extérieur connecté sur l'interface série). Les nouvelles versions de MS-DOS (version 3) ont introduit de nouveaux concepts, comme par exemple celui de l'impression en temps partagé : Vous envoyez à l'utilitaire PRINT le nom des fichiers à imprimer, et celui-ci, en arrière-plan, sans gêner le travail en cours, imprime les fichiers un par un. Il est possible sous MS-DOS comme sous CP/M de rediriger toutes les entrées sorties avec l'instruction MODE que nous avons vue plus haut (sorties d'imprimante parallèle sur le port série, sorties écran sur port parallèle, etc.). MS-DOS inclut également une routine de recopie d'écran graphique ou ASCII, c'est-à-dire que tout écran conservé en mémoire peut être vidé sur l'imprimante. Avec l'utilitaire GRAPHICS, la recopie est possible en haute résolution. MS-DOS insère également des utilitaires de reconfiguration de clavier pour divers pays du monde, KEYBFR pour la France, KEYBSP pour l'Espagne, KEYBUK pour le Royaume Uni, etc, etc.

Pour finir, sachez que MS-DOS est évolutif : initialement, il n'était pas même capable de gérer les disques durs. La dernière version commercialisée est la 3.2, la première version française qui équipe un micro-ordinateur (PC 1512). Elle a pour modifications notables le fait d'inclure des fonctions de gestion de réseau. Déjà la 4.0 est annoncée, et devrait permettre de faire fonctionner des logiciels d'arrière-plan comme DOS Plus. Et puis, l'on parle toujours de l'hypothétique version 5.0 qui serait multi-tâches, multi-utilisateurs, peut-être destinée aux ordinateurs comme le dernier 80386 de COMPAQ ?



OVERFLOW

La notion d'overflow concerne un dépassement. Prenons l'exemple d'un registre sur 16 bits soit une valeur de 65535 maximum, s'il est dépassé par le calcul, par exemple un calcul qui conduit à une valeur de 100 000, on aura alors un dépassement de capacité ou overflow. Ce type d'erreur peut générer un message de type OVERFLOW IN 10 par exemple qui signifie qu'un dépassement a été constaté dans un calcul en ligne 10.



OVERLAY

L'overlay est une technique de programmation qui permet de déporter le contenu exécutable d'un programme sur une mémoire de masse, et qui permet ensuite de le récupérer à un endroit spécifique de la mémoire pour le faire s'exécuter.

Il est ainsi possible de créer des programmes plus grands que la mémoire disponible, en rassemblant des fichiers exécutables dans des sous-fichiers. Dans un logiciel utilisant l'overlay (ou procédure de recouvrement), on réserve généralement un emplacement équivalent au plus important des blocs à charger. Une procédure d'overlay peut en règle générale utiliser elle-même plusieurs procédures d'overlay, et ceci ainsi de suite jusqu'à saturation des mémoires de masse. De nombreux outils de développement permettaient la gestion des overlays, lorsque la mémoire des ordinateurs était trop faible pour contenir l'intégralité des programmes à développer. Par exemple sous CP/M, il est impossible d'avoir plus 64 Koctets de mémoire utilisateur pour le fonctionnement des logiciels. On déporte donc un certain volume du logiciel sur disquette. Certains programmes sont ainsi des dizaines de fois plus volumineux que le corps principal apparent.
Il est généralement possible de repérer des procédures overlays sur une disquette en examinant le type des fichiers, un fichier d'overlay se terminant presque toujours par OVL. Des langages tels que Turbo Pascal offrent à l'utilisateur la possibilité de développer en overlay. Si vous détenez des programmes tels que DR DRAW, DR GRAPH, ou SUPERCALC, vous avez pu voir sur vos disquettes des fichiers se terminant par OVL. L'inconvénient de ce type de procédures est qu'elles occupent l'espace disponible sur les disquettes, en diminuant le volume de mémoire disponible sur les fichiers. Avec l'avènement des micro-ordinateurs de 512 K octets de mémoire voire 1024 K octets, la procédure d'overlay est devenue obsolète, et n'est pratlquement plus employée. Mais le monde informatique progressant à une vitesse impressionnante, il y a fort à parier que ces orgies de mémoire ne suffiront plus, et que les disques durs regorgeront bientôt de fichiers overlays dans tous les sens. C'est d'ailleurs ce qui commence à se passer, lorsque l'on s'aperçoit que certains logiciels nécessitent l'utilisation de huit ou dix disquettes.
Le second problème généré par l'overlay est le ralentissement général du programme qui les utilise, les accès disque étant toujours trop lents. Sur des ordinateurs tels que le PCW, le problème est partiellement résolu, puisque la présence du disque virtuel M offre à l'utilisateur une unité de pseudo-disquette, ayant un temps d'accès très proche de celui d'un accès mémoire dans un programme. Si un tel dispositif n'existe pas, le programmeur doit avoir recours à des ruses de sioux pour optimiser son logiciel au maximum, en regroupant dans un seul fichier OVL le maximum de fonctions devant être exécutées au même moment, ou en partitionnant la mémoire pour plusieurs fichiers de recouvrement.



PASCAL

Le langage PASCAL est beaucoup plus puissant que le BASIC à condition que l'on sache très bien le manier. Nous n'allons pas ici vous apprendre ce langage, mais simplement vous indiquer certains de ses avantages par rapport au BASIC, et qui décideraient peut-être ceux d'entre-vous qui hésitent encore à s'y mettre une fois pour toutes.

Tout d'abord, le PASCAL est, ce que l'on appelle, un langage structuré. C'est-à-dire que des instructions, comme le GOTO du BASIC sont prohibées. Un bon programme structuré en BASIC serait totalement constitué de blocs principaux et de sous-blocs ; ces derniers seraient appelés par les blocs principaux, par des GOSUB. En PASCAL, les blocs sont définis par des BEGIN et des END. On définit les sous-blocs grâce aux instructions PROCEDURE et FUNCTION qui, puisqu'elles définissent des sous-blocs, utilisent aussi les instructions BEGIN et END. L'autre grande particularité du langage PASCAL est sa récursivité. C'est un phénomène assez complexe à saisir lorsque l'on ne connaît que le BASIC, qui lui n'est normalement pas récursif. En BASIC, la récursivité serait, par exemple, la possibilité pour un sous-programme de s'appeler lui-même. C'est possible, me direz-vous ; mais dans une certaine limite car, je le répète, le BASIC n'est pas récursif. En PASCAL, il n'y a pas de limites. Cela permet de programmer en peu de lignes un algorithme complexe qui aurait pris beaucoup plus de place en BASIC.

Sur l'AMSTRAD, c'est le TURBO PASCAL qui est couramment utilisé. A l'automne 1988 Borland a hélas retiré ce très bon produit de son catalogue. Les inconditionnels du Pascal devront donc se rabattre sur d'autres marques. Il existe en Angleterre une version très normalisée du Pascal mise au point par Prospero Software ; il s'agit du Pro Pascal conforme à la norme ISO 7185.

Au catalogue des revendeurs français, on trouve plus facilement le Pascal MT+ de Digital Research ainsi que celui de HiSoft, un grand nom des langages de programmation. Vous pouvez même le compiler directement sur une disquette, plutôt qu'en mémoire, pour le réutiliser ensuite sous CP/M comme une commande CP/M.

N'ayez pas peur de vous mettre à ce langage, même s'il vous paraît très complexe car, avec un peu de volonté et beaucoup de patience, on arrive à tout. Le PASCAL existe sur AMSTRAD, il est plus rapide à l'exécution que le BASIC, alors il serait dommage de ne pas s'y intéresser.

PC

Evénement de l'année 1986, Amstrad lance un ordinateur compatible IBM PC. Présenté à la presse, à Londres la veille du P. C. W. Show (le "Sicob" anglais), les journalistes présents découvrent enfin celui dont on parle déjà depuis plusieurs mois. La machine est séduisante pour plusieurs raisons. Son prix d'abord. Ensuite parce que contrairement à beaucoup de compatibles, Amstrad livre un ordinateur avec deux systèmes d'exploitation, l'intégrateur GEM et GEM Paint, un puissant Basic et enfin, des utilitaires. En plus du logiciel, la configuration est de 512 Ko, carte couleur graphique, souris, 3 slots d'extensions, cartes série et parallèle, port joystick, écran monochrome ou couleurs, lecteurs de disquettes ou disques durs de 10 ou 20 Mo. Ces ensembles jamais vus encore sur le marché professionnel au moment du lancement de la machine, ne font pas que des heureux. Le PC 1512 est critiqué surtout sur sa compatibilité, sa solidité, ses possibilités. Pourtant le rapport qualité/prix est incomparable. La compatibilité est excellente. La solidité, nous verrons dans un an (le retour en atelier est très faible chez Amstrad).



Après avoir attaqué le marché de la bureautique avec ses PCW 8256 et 8512, Amstrad se lance sur le marché professionnel en proposant son compatible. L'abandon du sys-

tème de disquettes 3" pour le 5 1/4" sur ce matériel était nécessaire pour la compatibilité IBM. Mais cette entrée dans le milieu professionnel ne s'est pas fait seule. Un grand nombre d'éditeurs de logiciels proposent des versions de leurs produits "best", adaptées pour le PC 1512. Cette adaptation n'est pas, dans la quasi totalité des cas, dûe à une incompatibilité de la machine, mais à d'autres raisons. Il est évident que des éditeurs qui proposent des produits, bien souvent à plus de 5000F ne trouveraient pas de véritable marché chez les utilisateurs de PC 1512. Pour celà ils proposent des versions légèrement modifiées (au niveau des possibilités générales) de leurs meilleurs logiciels à des prix tournant autour de 1000F. L'acheteur d'un compatible de 20 000F ou 30 000F ne sera pas choqué par un logiciel coutant plus de 5000F. L'acheteur d'un PC 1512 sera plus hésitant. D'autres éditeurs ont aussi choisi simplement de baisser le prix des produits ayant déjà une longue carrière. Ils ne sont pas pour autant dépassés et l'utilisateur les retrouvera bien souvent sur son lieu de travail. Comme il n'existe aucun problème de compatibilité entre les versions pour PC 1512 et celles pour les autres compatibles, il est possible d'utiliser tous les fichiers déjà créés. L'Amstrad permet donc à un utilisateur de compatible IBM PC de mieux gérer son temps entre le bureau et les loisirs. Il lui est possible d'acquérir un PC 1512 pour un usage personnel, et de poursuivre un travail sans avoir à se déplacer vers son lieu de travail (toujours agréable le samedi ou le dimanche, n'est ce pas ?). Ce temps gagné lui permettra de mieux profiter de ses loisirs.

Le PC 1640 et son écran graphique amélioré marque une amélioration du produit. La grande surprise, ce fut la gamme PC 2000 présentée en septembre 1988 : vingt neuf machines positionnées en haut de gamme ! Les performances des 80286 et 80386 mises à la portée des petites entreprises. La sortie de ces machines avait été précédée d'accords d'échanges de technologies avec IBM : Amstrad faisait son entrée dans le club restreint des grands constructeurs d'ordinateurs.



Fiche technique des PC 1512.

Microprocesseur : 8086 à 8 Mhz.
Rom (mémoire morte) : 16 Ko.
Ram (mémoire vive) : 512 Ko extensible à 640 Ko (emplacement libre d'origine).
Mémoire de masse : Unité de disquettes 5 1/4" ou / et disque dur.
Systèmes d'exploitations : MS Dos (Microsoft), Dos Plus (Digital Research).
Interfaces d'origine : Centronics, Série, 3 slots d'extension type IBM PC, joystick, souris (livrée), carte multifonctions avec horloge (livrée), moniteur, unité de disquettes, disque dur, Ram supplémentaire.
Langage : Basic 2 (Locomotive Software).
Logiciels : Intégrateur GEM (Digital Research), GEM Desktop et GEM Paint (Digital Research), utilitaires de programmation et pour disque dur.
Graphisme :
Texte : moyenne résolution en 25 lignes de 40 caractères et 16 couleurs, haute résolution en 25 lignes de 80 caractères et 16 couleurs.
Graphique : Monochrome ou couleur. Résolution moyenne en 3 palettes de 4 couleurs sur 320 X 200 points, spéciale haute résolution en 2 couleurs sur 640 X 200 points et très haute résolution en 16 couleurs de 640 X 200 points.
Clavier : Azerty de 85 touches, dont pavé de fonctions et pavé numérique séparés.
Option : Imprimante DMP 3000 et périphériques spécifiques ou type IBM PC (Hard Card, tablette graphique, disque dur, carte graphique, etc.).
Documentation : Manuel complet sur la machine et ses logiciels. Manuel complémentaire (en option) sur Basic 2.
Configurations disponibles : Monochrome ou couleur, avec une ou deux unités de disquettes 5 1/4" ou une unité de disquettes et un disque dur de 10 ou 20 Mo.
Assistance : Service technique Amsoft (Sèvres) et Service Après Vente (France entière).

PCW

Après s'être attaqué avec succès à l'informatique familiale, Amstrad tente et réussit un coup de maître : imposer le traitement de texte pour tous. La machine à écrire prend un sacré coup de vieux à l'arrivee du PCW. Lequel s'offre le luxe d'être de surcroît un ordinateur de gestion très correct. Le PCW 8256 est destiné à la secrétaire, mais aussi aux journalistes, étudiants plongés dans une thèse ou un mémoire, bref aux gens qui écrivent. Le PCW 8512, avec sa mémoire plus étendue et son lecteur de 720 s'adresse plutôt aux professions libérales, artisans et commerçants qui lui confient leurs tâches de gestion.

La machine est livrée avec le système CP/M+, certes vieux mais performant. Le Basic Mallard est rapide mais dépourvu de commandes graphiques ; il existe heureusement des logiciels - comme Walbasic, de Waldata - qui pallient efficacement cette carence. A la place des graphismes, le Basic Mallard offre un outil très performant : le gestionnaire de fichiers Jetsam.
L'imprimante livrée avec le PCW fit le succès de la machine : finies les galères d'interfaçage ! Parsemer un texte d'enrichissements et de coquetteries de mise en page est un jeu d'enfant. L'imprimante matricielle, une Seikosha revue et corrigée, est un peu lente mais d'une qualité honorable.
Au fil des ans, le PCW s'est entouré d'une logithèque professionnelle : DBase II, Multiplan (un peu étriqué tout de même) et des logiciels verticaux ont été développés. Les jeux sont relativement peu nombreux mais d'une qualité parfois supérieure à leur équivalent sur CPC malgré l'absence de couleurs et de bruitage. La définition, il est vrai, est nettement à l'avantage du PCW.
Afin de se faire une place dans les bureaux des grandes entreprises, le PCW 9512 change de look. Carrosserie style PC, lecteur de disquette plus confortable (mais toujours en 3 pouces !), clavier plus ergonomique, il tente de séduire les patrons. Son atout : une large imprimante à marguerite qui garantit une frappe de très haute qualité. Un driver permet même l'interfaçage avec certaines imprimantes Laser, le fin du fin.

Locoscript 2 est doté d'un correcteur orthographique et d'une fonction mailing indispensable pour le courrier d'affaire.
Le PCW 9512 vise peu la clientèle privée (la série 8256/8512 avec l'imprimante graphique est plus adaptée) : c'est avant tout une machine de direction.

Les utilisateurs de PCW sont tous satisfaits de leur achat. La machine correspond à leur attente, elle remplit parfaitement son rôle. Une seule critique qui revient de temps en temps, le choix pour les lecteurs de disquettes en 3". Là, les fabricants indépendants ont détourné ce problème et proposent des lecteurs en formats 5 1/4" et 3 1/2" comme pour les CPC. Une chose se dégage toujours dans les discussions avec les utilisateurs de PCW : la facilité d'utilisation. Un petit détail qui a fait son importance au succès de cette machine.

Fiche technique des PCW 8256/8512

Microprocesseur : Z80A à 8 Mhz.
Rom (mémoire morte) : 256 octets.
Ram (mémoire vive) : 512 Ko.
Mémoire de masse : Unité de disquettes 3' (2 x 180 Ko formatés). Second lecteur 720 Ko pour le PCW 8512, en option pour le PCW 8256.
Système d'exploitation : CP/M Plus.
Langage : Basic Mallard, Logo.
Logiciel : Traitement de textes Locoscript (Locomotive Software).
Graphisme : Monochrome (noir et vert), 32 lignes de 90 colonnes.
Clavier : Azerty de 82 touches (plusieurs touches dédiées au traitement de textes et à la gestion de l'imprimante).
Documentation : Deux manuels complets sur la machine, la programmation du Basic et sur l'utilisation de Locoscript.

Fiche technique du PCW 9512

Microprocesseur : Z80A à 8 Mhz.
Rom (mémoire morte) : 256 octets.
Ram (mémoire vive) : 512 Ko.
Mémoire de masse : lecteur de disquettes 720 Ko formatés. Second lecteur en option.
Système d'exploitation : CP/M+ Langages : Basic Mallard, DR Logo. Logiciel : traitement de texte Locoscript 2.
Graphisme : monochrome (texte noir sur fond blanc), 32 lignes de 90 caractères.
Clavier : AZERTY 82 touches, dont certaines dédiées au traitement de texte et à la gestion de l'imprimante.
Option : Souris, tablette graphique, lecteur de disquettes, disque dur, etc.
Documentation : Deux manuels complets sur la machine, la programmation du Basic et sur l'utilisation de Locoscript.

PEEK & POKE

Le couple inséparable du basic et de l'informatique, poke et peek, peek et poke. La définition exacte de poke serait : affectation d'une valeur directe à la mémoire ; et celle de peek serait : lecture d'une valeur directe dans la mémoire.

Ces deux instructions sont les cordons ombilicaux entre le basic et la mémoire de l'ordinateur. Elles permettent de lire des données dans cette mémoire ou d'y écrire. Mais attention danger, la médaille a son revers. Si ses instructions permettent de lire et d'écrire dans la mémoire de bas en haut sans restriction, certaines adresses contenant par exemple du code ou des variables systèmes ne doivent pas être écrites sans faire très attention, ou l'on conduit la machine à un plantage inévitable et irrémédiable. Un grand nombre d'opérations sont réalisables avec peek et poke, et il est possible d'écrire une foule de logiciels utilitaires. La souplesse du Basic (ou de tout autre langage évolué acceptant ces instructions), permet d'effectuer des transferts dans des variables avec une ligne de type : a=peek 23679, qui aura par exemple pour effet de stocker le contenu de la case mémoire 23679 dans la variable numérique a. Ce petit exemple nous fait découvrir un point très important : les instructions PEEK et POKE n'ont pas la même structure.
PEEK est une fonction, c'est-à-dire qu'elle peut être placée derrière une variable ou dans une opération comme les exemples suivants : b=(PEEK 34000) -63, qui soustraira au contenu de l'adresse 34000 la valeur 63, et la placera dans la variable numérique B. Ou encore : PRINT PEEK 45000, qui affichera à l'écran le contenu de la case mémoire 45000. A l'inverse, l'instruction POKE n'est pas une fonction, mais bel et bien une instruction, c'est-à-dire qu'elle se place en début d'une commande de non derrière une instruction comme peek. Examinons la ligne suivante : POKE 2356, 64, qui disposera à l'adresse 2356 la valeur 64. Ou bien encore :
10 A=4
20 POKE 23606, A, qui placera à l'adrese 23606 la valeur contenue dans la variable A, c'est-à-dire 4. J'espère que vous avez bien saisi les différences fondamentales entre PEEK et POKE. Maintenant, vous devez pouvoir envisager foules d'applications utilisant ces instructions, comme par exemple le transfert d'un écran en mémoire. Ainsi, si nous disposons dans un CPC un écran à l'adresse 30000, nous devons utiliser une boucle BASIC qui implantera l'écran à l'adresse de la mémoire écran, c'est-à-dire l'adresse hexadécimale & C000. Les instructions peek et poke sont également fréquemment utilisées dans des programmes en assembleur pour stocker des données derrière une instruction DATA, puis les replacer en mémoire et enfin y accéder par une instruction CALL. Il est également possible de transférer des blocs de data à l'aide de sous-programmes très simples utilisant des boucles. Le seul inconvénient du peek et du poke est leur extrême lenteur quelle que soit le système, et il est bien souvent nécessaire de compiler le programme pour obtenir un résultat acceptable.

PLANTAGE

Le terme planter signifie, dans le jargon informatique, qu'à la suite d'une erreur de programmation (appelée aussi un bug), le système est bloqué et qu'il n'est plus possible pour l'utilisateur de "reprendre la main", et donc de retrouver l'accès au programme. Dans ce cas, la seule solution est d'éteindre, puis de rallumer l'ordinateur et de recommencer la programmation.

Ce problème arrive généralement lorsque l'on travaille en langage machine, ou lorsque l'on accède à certains endroits de la mémoire vive (RAM), ce qui peut perturber le fonctionnement de l'ordinateur sans pour autant détériorer le matériel. Les endroits où il convient d'éviter l'accès sont généralement ceux réservés au système d'exploitation. Par exemple, il ne faut pas imprudemment faire de POKE dans la zone mémoire &0000 à &016F car, si une partie de la ROM a été recopiée à cet endroit, ce

n'est pas pour rien. Le système d'exploitation a besoin d'y accéder très régulièrement quelle que soit la mémoire sélectionnée, ROM ou RAM. Lorsque l'on travaille en BASIC, à l'élaboration d'un programme (ou même à la recopie d'un programme se trouvant dans une revue comme celle-ci), le risque de plantage est très minime. Il arrive que par l' utilisation mauvaise d'un POKE ou d'un CALL. Pensez tout de même à ne mettre l'instruction ON BREAK CONT ou ON BREAK GOSUB qu'en dernier, c'est-à-dire lorsque le programme est au point (idem pour le ON ERROR GOTO). La raison en est simple : en cas de problème, si le BASIC n'interrompt pas le programme par un message d'erreur, vous ne pourriez pas l'interrompre vous-même par un BREAK. Dans le cas du langage machine, les initiés savent qu'il n'existe aucun message d'erreur. L'assembleur possède la caractéristique d'être un langage complexe au niveau de la compréhension, mais très limité dans ses fonctions de base. Donc, lorsque vous lancez un programme écrit en assembleur contenant la moindre erreur, il y a un risque de plantage. C'est un des langages les plus stricts qui, souvent, ne pardonnent pas la moindre faute. C'est pourquoi, lorsque vous programmez en assembleur, vous devez être prudent et TOUJOURS sauvegarder vos routines AVANT de les lancer. Il est toujours très attristant de devoir retaper un programme de deux pages à cause d'un plantage. Cette solution de sauvegarde doit aussi être employée lorsque vous tapez un programme BASIC contenant des codes machines en DATA. La moindre erreur dans ces lignes risquerait de planter le système lors d'un CALL. Ce procédé de sécurité évite de perdre inutilement des heures de travail, car dans le monde professionnel, chacun sait que le temps, c'est de l'argent. Une troisième cause du plantage est le phénomène de micro-coupure. Une micro-coupure est une défaillance de l'alimentation de l'ordinateur. C'est une coupure très brève du courant qui peut créer une perte des données contenues en RAM (les mémoires RAM perdent leur contenu lorsqu'elles ne sont plus alimentées), d'où un plantage du programme en cours. Lorsque vous devez taper un programme très long, pensez à faire une sauvegarde intermédiaire de temps en temps, une micro-coupure est si vite arrivée !

Soyez très prudents lorsque vous utilisez le langage machine, vous éviterez des heures de travail inutiles. N'oubliez pas qu'un plantage est sans remède pour récupérer les données contenues en mémoire vive.

PORT

Les ports sont des dispositifs spéciaux d'entrées-sorties utilisés par les microprocesseurs pour communiquer avec leur entourage ou leurs coprocesseur.

Il faut tout d'abord savoir lorsque l'on étudie un microprocesseur que celui-ci est toujours entouré sur une carte mère d'une multitude de coprocesseurs. Ces coprocesseurs sont en règle générale destinés à la gestion de l'écran, des interfaces séries, des interfaces parallèles, des contrôleurs de lecteurs de disquettes ou de disques durs, bref de toutes les composantes de base d'un micro ordinateur. Il existe divers modes d'adressage pour un processeur : celui-ci comprend tout d'abord plusieurs bus, qui sont des lignes sur lesquelles circulent des données. Il existe un BUS de commande destiné au contrôle des chips périphériques, un BUS d'adresse qui gère les adresses de la mémoire, et un BUS de données qui sert à transmettre des informations sur l'un ou l'autre des éléments du système. Parallèlement à tout cela, un dispositif permet au processeur d'adresser un certain nombe de cases mémoire spécifiques appelées ports. Ces ports servent à connecter des registres de coprocesseur afin de pouvoir réaliser un dialogue et de transmettre des données entre le processeur central et les processeurs annexes. Le nombre de ports varie selon les ordinateurs et les processeurs. Un processeur 8 bits ne gèrent généralement que 255 ports d'entrées-sorties (gestion des ports sur 8 bits donc), c'est les cas du 8080 et du Z80. les micro-ordinateurs comme le SINCLAIR SPECTRUM et le PCW8256 qui sont à base de processeur Z80 gèrent effectivement les ports sur 8 bits (255 adresses de ports possibles) mais un ordinateur de la famille des CPC, grâce à un GATE ARRAY spécifique (LE GATE ARRAY est un processeur spécialisé combinant plusieurs microprocesseurs, et crée par une société, dans le cas présent AMSTRAD), peut gérer 65535 ports d'entrées-sorties ; les processeurs de la

famille des 16 bits comme le 8088 ou le 8086 peuvent gérer d'origine et sans artifices 65535 ports d'entrées-sorties. Attention, si un port physique est toujours prévu pour des entrées et des sorties, les registres des processeurs qui lui sont connectés ne sont pas nécessairement de cet avis et sont alors configurés soit en entrée, soit en sortie. La gestion des ports se fait toujours par l'intermédiaire d'instructions spécialisées, que l'on retrouve dans tous les langages, et qui s'intitulent IN ou OUT. Ces instructions existent telles quelles en assembleur, comme en BASIC (BASIC LOCOMOTIVE, BASIC MALLARD, BASIC 2 compris). Ces instructions sont bien souvent accompagnées d'une autre qui s'appelle WAIT, et qui permet d'attendre une donnée à l'entrée d'un port d'entrée sortie spécifié.
Les instructions de gestion des ports sont souvent, peu ou pas utilisées par les programmeurs. Il est vrai que leur utilisation s'apparente à de la bidouille, et diminue la portabilité des logiciels. Pourtant les ports d'entrées sorties recèlent souvent des richesses insoupçonnables, et méritent qu'on les observe de plus près.

PPI 8255

Ce circuit est une interface parallèle qui a été développé à l'origine par INTEL pour le 8080. Il dispose de 24 canaux pouvant être utilisés comme entrées ou comme sorties. Ces 24 canaux sont divisés en trois ports de 8 bits qui sont les ports A, B et C. Chacun de ces ports a une fonction spécifique dans votre CPC. C'est ce que nous allons voir ensemble.

Le port A

Il est directement relié au PSG (circuit générateur de sons de l'AMSTRAD, qui est le AY-3-8912). Ce port permet la lecture du clavier et des manettes de jeu à travers le registre du R14 du générateur de sons ainsi que la communication entre ce dernier et le Z80.

Le port B

Il est utilisé pour lire les données en provenance du lecteur de cassettes. Il teste également le fonctionnement de l'imprimante connectée, grâce au signal BUSY. Il reçoit le signal VSYNC (signal d'interruption provenant du CRTC). le bit B4 sert à la sélection du réseau 50/60 Hz (50 hz en France, donc B4 est à 0). le bit B5 indique si une extension est connectée au CPC (B5 est relié à la borne EXP du bus d'extension). Les bits B3 à B1 sélectionnent le nom de l'ordinateur et sont, comme le bit B4, programmés directement à l'intérieur du CPC par des connexions sur la carte imprimée supportant les composants électroniques (en électronique, ce type de liaison s'appelle des straps).

Le port C

Il a cinq fonctions :

- sélection du PSG.
- gestion du mode de fonctionnement du PSG.
- sélection des lignes de la matrice du clavier par l'intermédiaire d'un décodeur 74 LS 145.
- écriture des données sur cassettes.
- contrôle du moteur du lecteur de cassettes (marche/arrêt).

Programmation du PPI

Pour programmer ce circuit, il vous faudra utiliser les instructions INP et OUT du BASIC, ou leurs équivalents en assembleur. Voici les adresses de ports où vous pourrez agir pour programmer le PPI et leurs fonctions respectives
&F4XX : lecture et écriture du port A. Donc accès au PSG.
&f5XX : lecture du port B (ce port est à lecture seule).
&F6XX : écriture dans le port C.
&F7XX : écriture dans le registre de contrôle.
Essayez de faire en BASIC un OUT &F600, &X00010000, sur un CPC464, vous mettrez le moteur de votre lecteur de cassettes en marche. Si vous posédez un autre CPC, l'effet obtenu sera le même à condition, de l'avoir connecté à un magnétophone, sinon, vous n'entendrez que le bruit du relais qui collera (un relais est un appareil utilisé en électronique ou en électrotechnique qui est une sorte d'interrupteur commandé par courant). Pour arrêter le moteur de ce lecteur de cassettes, il suffit tout simplement de faire un OUT &F600,0.

Nous venons ensemble de découvrir le circuit le plus simple, à utiliser dans le CPC. Ceux d'entre vous qui sont "bidouilleurs" pourront utiliser ce circuit pour leurs applications d'entrée et sortie de données.

PSG AY3 - 8912

Ce circuit est le générateur de sons programmables qui se trouve dans tous les CPC. Il est originaire de chez General Instruments et a été développé pour les jeux électroniques afin de les doter d'un son plus réaliste. Voyons maintenant comment il a été utilisé sur votre AMSTRAD.

Les concepteurs de ce circuit ont pensé qu'il serait souhaitable de pouvoir utiliser ce circuit directement avec un clavier ou un joystick. C'est pourquoi ce circuit possède un port 8 bits. Le PSG est composé des éléments suivants :

- des générateurs sonores : il y en a trois qui produisent un signal rectangulaire dont la fréquence est programmable. Ce sont les canaux A,B,C, lesquels sont indépendants entre eux.

- un générateur de bruit blanc : produit un son constitué par de nombreux signaux.

- un mélangeur : permet de mélanger les sorties des trois générateurs sonores et du générateur de bruit.

- un contrôleur d'amplitude : il sélectionne l'amplitude de sortie du signal, ce qui a pour effet de fixer le volume sonore.

- un générateur d'enveloppe : il possède huit formes d'enveloppes et produit une enveloppe de modulation d'amplitude.

- convertisseurs digitaux-analogiques : au nombre de trois, ils produisent des signaux pouvant prendre seize valeurs, déterminées par le contrôleur d'amplitude.

- port d'entrée/sortie : il n'est pas utilisé pour la production de sons, mais il s'occupe de la lecture du clavier et de la manette de jeux.

les registres du PSG

l'AY-3-8912 dispose de seize registres dont quinze peuvent être utilisés. Ils permettent de programmer toutes les possibilités sonores du circuit. Voici les fonctions de ces registres :

- Reg. 0,1 : tonalité du son sur le canal A. Les huits bits du registre 0 et les 4 bits inférieurs du registre 1 sont utilisés. Plus la valeur représentée par ces douze bits est petite, plus le son sera aigüe.
- Reg. 2,3 : comme Reg 0,1 pour le canal B.
- Reg. 4,5 : comme Reg. 0,1 pour le canal C.
- Reg. 6 : influence le générateur de bruit avec ses cinq bits inférieurs (32 combinaisons possibles).
- Reg. 7 : registre multifonctions.
- Reg. 8 : les quatre bits inférieurs de ce registre fixent le volume sur le canal A. Si le bit B4 est à 1, le contenu des bits 0 à 3 est ignoré et le volume est alors déterminé par le registre de courbe d'enveloppe.
- Reg. 9 : comme Reg. 8 pour le canal B.
- Reg. 10 : comme Reg. 8 pour le canal C.
- Reg. 11,12 : les seize bits de ces deux registres influencent la période de la courbe d'enveloppe.
- Reg. 13 : les bits 0 à 3 de ce registre déterminent la forme de la courbe d'enveloppe.-
Reg. 14 : il gère le port d'entrées/sorties qui s'occupe de la lecture du clavier et de la manette de jeux. Le bit B6 du registre R7 règle le sens de la transmission, mais comme le port est utilisé en entrée, il suffit de mettre ce bit à 0.

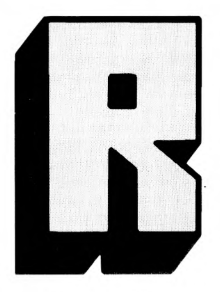
Le registre 7 étant très important, voyons la fonction de chacun de ses bits :

B0 : mettre/couper le son du canal A (0 = mis / 1 = non).
B1 : même chose pour le canal B.
B2 : même chose pour le canal C.
B3 : mettre/couper le bruit sur le canal A (0 = mis / 1 = non).
B4 : même chose pour le canal B.
B5 : même chose pour le canal C.
B6 : port A comme entrée/sortie (0= entrée / 1 = sortie).
B7 : même chose que b6.

Programmation du PSG

Nous avons vu que le PSG n'est pas directement relié au bus du Z80. Sa programmation se déroule en quatre étapes :
1) Introduction sur le port A du PPI de l'adresse du numéro de registre du PSG à programmer.
2) Introduire, sur les deux bits de poids fort du port C (B6 et B7) , la valeur 11 est nécessaire pour adresser le PSG. L'adresse du registre du PSG est alors chargée.
3) Introduire, sur le port A du PPI, la valeur à charger dans le registre sélectionné.
4) Postionner le PSG en mode écriture en introduisant la valeur 10 sur les bits 6 et 7 du port C du PPI.

Nous venons de voir un circuit qui n'est pas des plus évident à programmer, mais avec un peu de pratique, vous arriverez certainement à faire des sons assez surprenant en programmant directerment le PSG sans passer par les instructions sonores du BASIC.

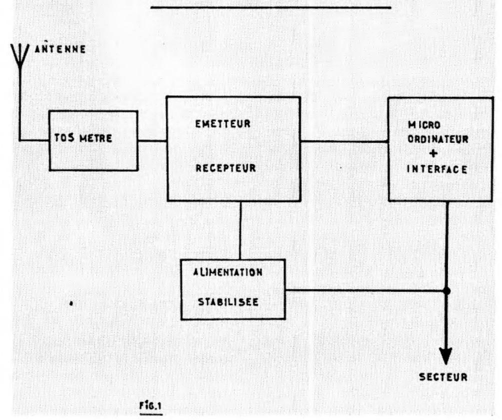


RADIO ASSISTÉE PAR ORDINATEUR

Les radio-amateurs utilisent de plus en plus les micro-ordinateurs lors de leurs communications. Ceux-ci sont le plus souvent utilisés comme décodeurs de signaux tels les messages en code morse.

- Il existe plusieurs modes d'émission :
FM, AM, USB, LSB, CW, RTTY.
- FM = modulation de fréquence : utilisée par les radios libres. Le signal est très clair à sa réception, mais il ne porte pas très loin.
- AM = modulation d'amplitude : utilisée par les transmissions de radio à grand taux d'écoute comme les radios nationales. Lorsque vous écoutez une émission sur les grandes odes ou sur les petites ondes, votre poste de radio reçoit des signaux transmis en AM.
- USB et LSB = respectivement bande latérale supérieure et bande latérale inférieure. Souvent appelées sans distinction BLU (Bande Latérale Unique). Il s'agit de ne transmettre que l'enveloppe du signal et suivant le mode USB ou LSB utilisé, on garde seulement l'enveloppe positive ou négative du signal.
- CW = morse : il s'agit simplement de transmettre des impulsions.
- RTTY : radio télétype : comparable à un télex par liaison radio. Il permet de transmettre des messages sur écran. Les caractères sont comme en informatique, codés en ASCII.

En FM, AM et BLU, il n'est bien sûr pas nécessaire de décoder les signaux à la réception par l'intermédiaire d'un micro-ordinateur (ni bien sûr à l'émission) puisque l'on transmet la voix humaine. Par contre, dans le cas de la CW ou de la RTTY cela peut être utile (même obligatoire dans le cas de la RTTY). Le problème majeur du décodage morse réside dans la synchronisation entre les stations en liaison radio. En effet, l'ordinateur possède un signal d'horloge qui a une fréquence fixe et précise. Il est facile de prendre un repère temporel dans un programme grâce à certaines fonctions ou adresses mémoires prévues pour cela (voir les fonctions TIME, AFTER et EVERY du CPC). Un humain, par contre, ne possède pas d'horloge interne. Donc, lorsqu'il tape un message en morse sur son manipulateur, il ne le fait pas à vitesse constante d'où une possibilité d'erreur lors du décodage. Le cas idéal est celui d'une liaison avec codage et décodage par ordinateurs entre les stations en liaison. Dans ce cas, la synchronisation est simple à obtenir et les risques d'erreurs de réception résident surtout dans les interférences radios. Les utilisateurs n'ont plus qu'à taper leurs messages au clavier de leur micro-ordinateur qui se chargera de les trans-



mettre en morse par l'intermédiaire de l'émetteur auquel il est relié.
Ce procédé d'utilisation est strictement le même pour la RTTY. La différence réside dans le fait qu'il faille obligatoirement un ordinateur dans la station émettrice et un dans la station réceptrice qui codera et décodera tous les messages.
Une autre utilisation des micro-ordinateurs par les radio-amateurs est celle qui est purement logicielle tels les programmes de localisation, d'apprentissage du code morse, de répertoire ...
La figure 1 montre la configuration minimum nécessaire pour utiliser un micro-ordinateur avec une station de radio-émetteur. Le TOSMETRE est un appareil permettant de régler l'antenne en fonction de l'espace qui l'entoure et de la fréquence à laquelle elle travaille pour être le mieux accordée possible avec l'émetteur (un mauvais réglage de l'antenne peut avoir des effets néfastes sur l'émetteur).
Nous vous conseillons, si vous êtes concerné et possesseur d'un AMSTRAD CPC, l'ouvrage "COMMUNIQUEZ AVEC AMSTRAD" de D. BONOMO et E. DUTERTRE.

E. MISTELET

RÉSEAU

Un réseau est un procédé qui permet à plusieurs ordinateurs d'être reliés entre eux, et de partager des ressources communes telles qu'une imprimante ou un disque dur. Il existe plusieurs types de réseau dont les architectures sont connues et servent maintenant de référence pour tout type de développement : les réseaux en anneau, les réseaux en étoile, les réseaux en bus, les réseaux maillés, et les réseaux en arbre.

Un réseau en anneau est une boucle sur laquelle viennent se raccorder des micro-ordinateurs, cette boucle se terminant toujours en se connectant à elle-même ; ce type de connexion présente pour avantage la bonne circulation des informations et pour désavantage ses difficultés d'implantation ; les réseaux en étoile connectent toutes les unités à un noyau central qui regroupe toutes les connexions ; l'avantage d'une telle solution est l'interconnexion totale de tous les éléments du réseau, le désavantage est dû à la centralisation dans un noyau, des informations. Les réseaux sur bus sont constitués par une ligne parcourant un espace, sur laquelle viennent se connecter diverses unités. L'avantage de cette solution réside dans sa grande facilité d'implantation (solution choisie par des réseaux de type ETHERNET). Les deux dernières solutions sont le réseau maillé, idéal pour le transfert d'informations, puisque plusieurs ensembles de données à destination de différents postes peuvent circuler en même temps, mais complexe car ayant tendance à se transformer en spaghetti.
Le réseau en arbre est constitué d'une structure arborescente, mais pose toujours le problème de l'origine par laquelle doivent le plus souvent transiter les informations. La solution du réseau local est une manière de plus en plus fréquemment prisée par les sociétés pour s'informatiser, car elle est évolutive, et ne nécessite par l'achat d'un gros ordinateur central relié à des terminaux, qui bien souvent doit être changé dès que l'on désire passer à la vitesse supérieure.
Dans un réseau, pas de problème, les ressources sont partagées par une multitude de postes de travail réfléchissant chacun de son côté.
On peut avec un réseau partager un ou plusieurs disques durs entre plusieurs utilisateurs, voir même des lecteurs de disquettes, ou des imprimantes comme par exemple des imprimantes à laser. A l'heure actuelle, les solutions disponibles sur le marché sont encore chères, mais certains constructeurs tels la société NORTHERN COMPUTER commencent à proposer des systèmes à faible prix sans pour autant offrir de perfomances trop limitées. On commence même à voir des réseaux fonctionnant en RS232. Des logiciels commencent à prévoir des fonctions de réseau (DBASE III + d'ASHTON TATE). En effet, pour qu'un logiciel puisse fonctionner en multi-utilisateurs, il suffit qu'une option lui permette de travailler avec des fichiers vérouillés (cas du JETSAM de BASIC MALLARD par exemple, qui fonctionne très bien avec le réseau AMSTORE de la société MERCI).

ROM DES CPC

Les CPC, outre leurs 64 K de RAM, sont équipés de plusieurs ROM's qui contiennent les éléments indispensables à la marche du micro-ordinateur. Ces ROM's sont au nombre de deux pour le CPC 464, et de trois pour les CPC 664 et 6128. Un CPC 464 équipé d'un lecteur de disquettes peut être considéré com-me ayant autant de ROM's qu'un 664 ou un 6128.

Un microprocesseur 8 bits semblable au Z80 qui équipe les CPC ne peut adresser qu'un maximum de 64K de mémoire. Cet espace étant déjà intégralement occupé par la RAM des CPC, il a été nécessaire de recourir à la technique du "bank switching" pour inclure les diverses ROM's dans l'espace adresse du Z80. Cette technique est purement logicielle et consiste à recouvrir virtuellement un intervalle de RAM par le contenu d'une ROM. Le résultat de cette opération n'altère pas le contenu de la RAM ainsi recouverte, mais fait en sorte que toutes les lectures auront lieu dans la ROM à laquelle ont été allouées les adresses "commutées".
Dans l'espace adresse du Z80 des CPC, les possibilités sont multiples. La zone comprise entre les adresses &0000 et &3fff a été allouée à la ROM du système d'exploitation. La zone entre &c000 et &ffff est utilisée par la ROM du Basic et pour celle du système de gestion des disquettes (ROM dans laquelle est incluse une partie du LOGO). Toutefois, cette dernière zone peut aussi être partagée avec d'autres ROM's que celles des CPC, et est d'ailleurs utilisée par les diverses extensions connectables à ces ordinateurs (Extensions du type VORTEX, MULTIFACE TWO, SILICON DISK ou autres). Compte-tenu des possibilités offertes par le "bank switching" et de la structure des CPC, il est possible de connecter dans cette zone jusqu'à 252 ROM's supplémentaires ce qui porte l'espace adresse du Z80 à près de 4 MO.
Cette capacité maximale est purement théorique et est, en réalité, limitée par le temps de commutation et les nécessités d'alimentation des diverses extensions.
Chaque ROM située dans les adresses &c000 à &fff est considérée comme une ROM d'extension et doit commencer par quatre octets destinés à permettre son identification. Ces quatre octets sont suivis par l'adresse de la table des commandes (ou instructions) que contient cette ROM. Une ROM occupe toujours de la place en RAM. Cette place est au minimum de quatre octets destinés à permettre sa reconnaissance par le système, et peut être étendue à volonté en fonction des nécessités de sauvegarde de valeurs que demande la mise en œuvre de la ROM (une ROM ne pouvant être accessible qu'en écriture, il sera nécessaire de stocker en RAM les valeurs intermédiaires utilisées par ses routines). Ces contraintes sont incontournables et vous devrez les respecter si vous voulez créer vos propres ROM'S.
Il peut être intéressant de connaître de façon détaillée le contenu des ROM's. Celui-ci est d'ailleurs entièrement décrit dans des ouvrages spécialisés. Mais ce contenu peut être modifié d'une version des CPC à une autre et cette possibilité de différences ne permet pas de dire qu'un logiciel faisant directement appel à des adresses ROM sera compatible entre plusieurs CPC différents. Toutefois AMSTRAD a prévu, pour chacune des versions du CPC, une table de sauts standards (ou JUMPBLOCK) résidente en RAM et permettant, et ce quelle que soit la version du CPC utilisée, d'accéder à certaines routines des ROM's réalisant des fonctions particulières. Seule l'utilisation de ce JUMPBLOCK permet d'assurer la compatibilité entre deux modèles de la série des CPC. Ce JUMPBLOCK permet d'activer, essentiellement en assembleur Z80, des fonctions généralement inaccessibles à partir du Basic standard et pourtant très utiles. Voici une liste des principales adresses d'appel ainsi que leurs effets et leurs conditions d'appel.
&BB48 : interdit les interruptions par BREAK.
Cette routine est très intéressante pour qui désire interdire la possibilité d'interrompre un programme par la touche BREAK.
&BBCF : positionne les coordonnées gauche et droite de la fenêtre graphique. Les paires de registres DE et HL contiennent les coordonnées standard des bords gauche et droit de la fenêtre graphique.
&BBD8 : positionne les coordonnées haute et basse de la fenêtre graphique. Les paires de registres DE et HL contiennent les coordonnées standard des bords haut et bas de la fenêtre graphique.
&BBDB : efface la fenêtre graphique dans la couleur du PAPER graphique.
Ces trois routines sont très utiles pour effectuer un mélange de texte et de graphique dans un traitement. En

effet, les tracés graphiques ne sortiront jamais des limites allouées à la fenêtre.
&BBDE : fixe la valeur du PEN graphique.

Le registre A doit contenir le numéro de l'INK affectée au PEN graphique.
&BBE4 : fixe la valeur du PAPER (fond) graphique. Le registre A doit contenir le numéro de l'INK affectée au PAPER graphique.
&BC44 : colore une aire rectangulaire avec la couleur précisée. Cette fonction est limitée à des coordonnées données en mode caractère. Le registre A doit contenir le numéro de l'encre (masquée) de remplissage. Les registres H et D contiennent respectivement les numéros des colonnes droite et gauche de la zone à remplir. Les registres L et E contiennent respectivement les coordonnées haute et basse de la zone.
&BCD1 : permet de faire reconnaître par le système des RSX logées en RAM. La paire de registres BC doit contenir l'adresse de la table des commandes RSX à inclure. La paire des registres HL doit contenir l'adresse.
&BCD4 : recherche une RSX dans les ROM's connectées.

La paire de registres HL doit contenir l'adresse.
&BD19 : synchronise l'affichage avec le retour du spot de l'écran. Cette routine permet d'éviter le clignotement de l'affichage lors du déplacement de symboles ou graphismes sur l'écran.
D'autres adresses standard correspondent à des fonctions particulières. Notre but n'étant pas ici de vous présenter l'intégralité des possibilités du JUMPBLOCK standard, nous allons terminer par la présentation de quelques routines de la ROM de gestion des disquettes. Celles-ci sont accessibles avec un 664, un 6128, ou un 464 avec lecteur de disquette DDI-1. Au préalable il est bon que vous sachiez que, pour l'ensemble de ces routines, les conditions d'appel sont les suivantes :
- registre E = numéro d'unité, soit 00 pour le lecteur A et 01 pour le lecteur B ;
- registre D = numéro de piste ;
- registre C = numéro de secteur ;
- paire HL = d'une zone de 512 octets contenant le secteur concerné par l'opération.

&BE89 : lecture d'un secteur de la disquette.
&BE8C : écriture d'un secteur sur la disquette.
&BE8F : formatage d'une piste entière. Dans ce cas, la paire HL contient l'adresse d'une table qui précise les caractéristiques des différents secteurs.
Ces routines ne sont pas accessibles sous AMSDOS. Pour pouvoir profiter de leurs possibilités, vous devrez utiliser leur codification sous forme "d'ordres Basic cachés". Ces ordres, que vous ne trouverez pas dans le manuel utilisateur du DDI-1, sont les suivants :
- &81 : interdiction / autorisation des messages d'erreur disquette.
- &82 : patch des paramètres de disque.
- &83 : sélection d'un format de disquette.
- &84 : lecture d'un secteur.
- &85 : écriture d'un secteur.
- &86 : formatage d'une piste.
- &87 : positionnement de la tête sur une piste particulière.
- &88 : permet de déterminer si un lecteur est disponible.
- &89 : précise le nombre de tentatives de lecture avant message d'anomalie.

Sous AMSDOS, ces ordres sont accessibles par l'utilisation de la routine qui est appelée par un CALL à l'adresse &BCD4.
Nous venons de vous donner un bref résumé du contenu et des possibilités des ROM's standard des CPC. Pour avoir des renseignements complets sur cette partie de votre micro-ordinateur, vous aurez tout intérêt à vous reporter aux ouvrages suivants.
- COMPLETE FIRMWARE (SOFT 158) d'AMSTRAD,
- DDI -1 FIRMWARE (SOFT 158A) d'AMSTRAD,
- LA BIBLE DU PROGRAMMEUR DE L'AMSTRAD CPC de MICRO-APPLICATION,
- LE LIVRE DU LECTEUR DE DISQUETTES AMSTRAD de la MICRO-APPLICATION.

R.P SPIEGEL

ROS

Nous avons déjà expliqué ce qu'est un BIOS à la lettre B. Oui me direz-vous mais quel rapport ? Et bien c'est tout simple : en fait, avant le mois de septembre de cette année, le terme ROS n'existait pas, ou alors n'était cité que dans des salons très fermés. ROS est tout simplement le nom inventé par AMSTRAD pour désigner le BIOS de son PC. ROS signifie ROM OPERATING SYSTEM. Pourquoi pas, AMSTRAD innove encore, à la limite, le terme serait même plutôt mieux adapté que BIOS. Alors désormais, soyez branchés, dites ROS et non BIOS. Personnellement je commence dès maintenant.

ROUTINE

Une routine est une suite d'instructions dans n'importe quel langage (on peut parler de routines en PASCAL, en assembleur, en BASIC etc.). Cette suite d'instructions exécute généralement une opération spécifique utilisée par un programme. Un programme

utilise plusieurs routines, et même des sous-routines (appelées par l'instruction CALL ou GOSUB, et rendant la main au programme par un RET ou RETURN). A ne pas confondre avec les procédures qui peuvent également être des routines après tout, voir même composées de routines. D'ailleurs une routine peut elle-même être composée de routines et de sous-routines (qui, pourquoi pas, contiendraient elles-même des routines ; mais alors, il ne s'agit plus d'une routine mais d'un programme ou d'un sous-programme ?). Il est vraiment difficile de décrire précisément ce qu'est une routine.



Il est toujours intéressant de pouvoir transférer des informations d'un ordinateur à un autre, et c'est pour cela que les interfaces série furent créées. Car en fait, le nom RS232 n'est qu'un nom de norme, qui est regroupé avec beaucoup d'autres dans la famille des interfaces de transmission en série.

Qu'est-ce qu'une transmission en série ? Lorsqu'un ordinateur dialogue avec des périphériques, il dialogue généralement (surtout au niveau de sa carte principale) en parallèle. C'est-à-dire qu'il utilise un bus de transfert, représenté par plusieurs lignes sur lesquelles transitent des informations. Ces lignes sont soit au nombre de 8, soit au nombre de 16. On parle alors de transfert sur 8 ou 16 bits. Mais dès que l'on désire transmettre des informations sur une distance importante (plusieurs mètres ou dizaines de mètres), la liaison sur bus en parallèle devient trop onéreuse pour être choisie. On a alors inventé le concept de liaison en série, c'est-à-dire une seule ligne transférant les bits les uns à la suite des autres. C'est le cas des interfaces RS232. L'avantage de ce type de liaison est qu'elles permettent de transférer des informations à peu de frais sur plusieurs dizaines de mètres (voire plusieurs kilomètres avec un modem et une ligne téléphonique), l'inconvénient est qu'elles soient 8 fois plus lentes qu'une liaison parallèle sur huit bits ou 16 fois plus lentes qu'une liaison parallèle sur 16 bits, voire même dans certaines situations beaucoup plus. Les interfaces RS232 sont présentes sur tous les modèles d'ordinateurs AMSTRAD. Sur les CPC avec l'interface référencée RS232, sur les PCW 8256 8512 avec la CPS8256, et sur tous les PC.

Les configurations

L'interface RS232 offre de nombreuses possibilités de configuration. Elle dispose de nombreuses vitesses (50, 75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200), cette vitesse étant toujours exprimée en "Bauds" (nombre de bits transmis en seconde). Cette vitesse peut généralement être exprimée en entrée et en sortie indifféremment, afin de permettre la connexion d'appareils tels que les modems pouvant avoir des vitesses d'émission et de réception différentes (1200/75 par exemple, qui correspond à TELETEL). Il est également possible de configurer le contrôle de parité (NOME, EVEN, ODD soit : aucune, paire, impaire), pour demander à l'interface de réaliser un contrôle de validité des données reçues. Sont configurables également le nombre de bits de stop (bits reçus pour spécifier un arrêt), et le nombre de bits de données (6, 7 ou 8) qui vont permettre de rationaliser au maximum la transmission en ne transmettant que la série de bits correspondant au type de caractères demandés, (Par exemple, si vous vous contentez d'envoyer des codes ASCII inférieurs à 128, vous gagnerez du temps sur la transmission en envoyant les données sur 7 Bits. Pour finir, le contrôle de déroulement ou handshaking, est un mode de régularisation du flux des données par hard, c'est-à-dire que l'interface va contrôler la transmission avec l'aide de l'une de ses lignes de transmission. Tous ces modes qui peuvent parfois sembler complexes vont permettre à l'utilisateur de réaliser une transmission la plus efficace possible en fonction de ce qu'il veut faire. L'interface RS232 est caractérisée par plusieurs broches (8 en tout), qui ont chacune une fonction très précise. Tout d'abord les broches TX et RX qui servent à la transmission et à la réception de données. Elles sont généralement connectées aux deux autres broches TX RX de l'appareil relié. Puis la broche RTS (Ready to Send) qui indique au poste receveur que l'émetteur est prêt pour envoyer des données. La broche CTS (Clear to send), généralement reliée à RTS, est le signal inverse ; il signifie d'accord pour en-

voyer. Il est validé lorsque le récepteur est prêt à recevoir les données de l'émetteur.

Le signal GND donne la masse, et permet généralement à deux interfaces de sources différents (constructeurs d'interfaces, ou processeur de gestion de transmission différents) d'aligner leurs tensions. Ce signal est très important, car si deux circuits électroniques n'ont pas exactement les mêmes tensions, ce qui est fréquent, les différences de niveau s'estomperont. La broche DCD (Dat Carier Detect) signifie détection du support de données, et sert à gérer le protocole de transmission. La broche DTR (Data Terminal Ready ou terminal de données prêt) sert à gérer le flux de données entre le récepteur et l'émetteur. Si cette broche n'est pas connectée, les informations vont être envoyées sans arrêt, et il est possible qu'à un moment ou un autre le récepteur soit saturé, et des caractères risquent de sauter pendant la transmission. Avec DTR, la transmission est automatiquement arrêtée quand le buffer du récepteur ne peut plus accepter de donner, et repart quand de nouveaux caractères sont acceptables. Pour finir, la broche RING ou détecteur de sonnerie lorsqu'elle est utilisée avec un modem, indiquera à l'appareil connecté que le téléphone a sonné, afin que celui-ci déroche la ligne et commence le protocole de connexion.

Les interfaces séries ont de nombreuses utilisations. Elle nous permettent de transférer des fichiers de texte ou binaire entre deux machines lorsque, par exemple, des formats de disquettes sont différents. C'est cette méthode qu'a par exemple utilisé l'association OUF pour transférer ses utilitaires du domaine public de disquette au format 5" 1/4 vers les disquettes AMSTRAD 3".

Une interface série peut également servir à connecter une multitude d'imprimantes (la majeure partie des imprimantes à marguerite fonctionne généralement en série), et surtout de modems, qui nous permettront l'accès aux banques de données par TRANSPAC, ou sur TELETEL (le réseau des minitels). Néanmoins, le fait de posséder une interface série ne résoud pas tous les problèmes de transmission, sachez qu'elle doit toujours être accompagnée d'un logiciel approprié pour réaliser une connexion, et que les problèmes de connectique qu'elle pose parfois peuvent êtres insolubles pour des néophytes.

RSX

Sous ces trois lettres se cache une des possibilités les plus intéressantes de votre cher CPC. RSX signifie en anglais, RESIDENT SYSTEM EXTENSION ce qui donne en français EXTENSION RESIDENTE DANS LE SYSTEME.

description, fonctionnement et exemples

Vous avez déjà dû remarquer que le Basic de l'Amstrad possède un jeu d'instructions très complet. Grâce au RSX, vous pouvez vous-même en rajouter. Il vous faut pour cela posséder un minimum de connaissances en langage machine ainsi que certaines adresses utiles de votre CPC. Les possesseurs d'un Amstrad munis d'un lecteur de disquettes utilisent très fréquemment des instructions RSX telles que REN, ERA, DIR, CPM... Elles doivent toujours être précédées de la barre verticale I qui s'obtient en appuyant simultanément sur SHIFT et @. Cette barre indique à l'ordinateur que le mot qui lui est associé doit être considéré comme une instruction Basic. Celle-ci doit bien sûr avoir été précédemment préprogrammée. Si vous tapez 1 RECTANGLE juste après avoir allumé votre ordinateur, vous obtiendrez le message d'erreur UNKNOWN COMMAND qui vous signalera que la fonction RECTANGLE n'a pas été définie.

La définition d'une instruction RSX est plus simple que l'on peut l'imaginer. Il faut tout d'abord signaler à l'ordinateur la présence de cette instruction. Il faut ensuite définir son nom et bien sûr sa fonction. C'est cette dernière étape qui posera le plus de problèmes à beaucoup d'entre vous puisqu'elle nécessite la connaissance de la programmation en langage machine.

Le plus simple pour vous si vous désirez créer une RSX est de posséder un assembleur et bien sûr de savoir l'utiliser. La principale adresse à connaître est &BCD1 qui permet d'introduire une RSX. Deux conditions obligatoires lors de l'appel de cette routine : la paire de registres BC doit contenir l'adresse de la table de commandes RSX et la paire de registres HL doit contenir l'adresse d'une zone mémoire libre de quatre

octets (en RAM) qui sera utilisée lors de l'appel de la fonction. Ce qui nous donne la routine :

ORG # 9000 ; position en mémoire de la routine

TAMPON DEFS 4 : réservation d'une zone de 4 octets.

LD BC,VECT

LD HL, TAMPON

CALL # BCD1 : appel de la routine KL LOG EXT

RET : fin de la routine de définition.

VECT. DEFW TABLE ; définition de l'adresse de la table.

JP routine : sauf à la routine en langage machine.

TABLE DEFM : nom de l'instruction sans la dernière lettre.

DEFB # 80 + : "code ASC II de la dernière lettre du nom + # 80.

DEFB 0 : fin de la table des noms.

ROUTINE : début de la routine en langage machine que vous devrez créer.

Cette routine sera positionnée en &9000 (ORG 9000), donc, vous devrez valider la nouvelle commande RSX en tapant, depuis le Basic, CALL &9000. Ce n'est qu'un exemple. Vous pouvez très bien la placer où vous voulez. Vérifiez toutefois qu'elle ne vienne pas se loger à des endroits réservés comme dans la table de vecteurs par exemple (les vecteurs sont des adresses RAM permettant d'accéder à des routines ROM par des restarts RST).

Attention, le nom de la fonction doit être obligatoire en majuscule car toutes les instructions BASIC que vous tapez au clavier sont automatiquement transposées en majuscules.

Vous pouvez définir plusieurs commandes RSX en une seule fois. Supposons les commandes 1 RECTANGLE, 1 RDSECT, 1 WRSECT. Pour les définir, il suffit de modifier la fin de la routine précédente :

VECT DEFW TABLE

JP rectangle.

JP rdsect

JP wrsect

TABLE DEFM RECTANGL

DEFB # 80 + "E"

DEFM RDSEC

DEFB # 80 + "T"

DEFM WRSEC

DEFB # 80 + "T"

DEFB 0 ;

fin de la définition de la table rectangle ....;

début de la routine rectangle.

Comme vous pouvez le constater, il suffit, pour définir plusieurs commandes, de les programmer les unes après les autres en respectant le même ordre dans la table des vecteurs (table des JP) que dans la table des noms.

Vous avez peut-être déjà remarqué que certaines commandes RSX comme USER demandent des paramètres, (USER, 3 par exemple). La prise en compte de ceux-ci est très simple. Lors du lancement d'une commande RSX, certains registres contiendront des valeurs que vous pourrez prendre en compte dans vos routines. Le registre A contient le nombre de paramètres qui accompagnent la commande RSX à exécuter. Il suffit donc de mettre en début de routine l'instruction CP suivie d'une valeur pour tester si le nombre de paramètres est convenable. Par exemple, si votre commande doit être accompagnée de deux paramètres, vous devez faire : CP # 2 ; teste si deux paramètres RET NZ ; retour si nombre de paramètres incorrect.

Il reste maintenant à connaître la valeur des paramètres. Le registre IX est là pour ça. IX pointe sur l'adresse basse du dernier paramètre de la commande et IX + 1 sur l'adresse haute de celui-ci. De même, IX + 2 pointe sur l'adresse basse du deuxième paramètre et IX + 3 pointe sur l'adresse haute de celui-ci. Chaque pa-

```
        ORG  #9000
tampon  DEFS 4
        LD   hl,tampon
        LD   bc,vect
        CALL #bcd1
        RET
vect    DEFW table
        JP   rdsect
        JP   wrsect
table   DEFM RDSEC
        DEFB #80+"T"
        DEFM WRSEC
        DEFB #80+"T"
        DEFB 0
rdsect  CP   #2                  ; teste si 2 parametres
        RET  nz                  ; retour sinon
        LD   a,#84               ; code instruction lecture
        LD   (inst),a
        JR   dskprg              ; saut a'la routine principale
wrsect  CP   #2                  ; teste si 2 parametres
        RET  nz                  ; retour sinon
        LD   a,#85               ; code instruction ecriture
        LD   (inst),a
dskprg  LD   a,(ix+0)
        LD   (sect),a
        LD   a,(ix+2)
        LD   (piste),a
        LD   hl,inst
        CALL #bcd4               ; call KL-FIND-COMAND
        LD   (adr),hl
        LD   a,c
        LD   (adr+2),a
        LD   e,0                 ; No du lecteur 0=A/1=B
        LD   a,(piste)
        LD   d,a                 ; No de piste
        LD   a,(sect)
        LD   c,a                 ; No du secteur
        LD   hl,#9500            ; adresse du buffer
        RST  #18
        DEFW adr
        RET
inst    DEFS 1
adr     DEFS 3
piste   DEFS 1
sect    DEFS 1

Text:#68F5        End:#6B53         # 25E Bytes
Hmem:#8D39
```

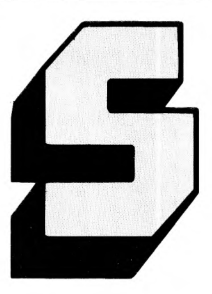
ramètre est donc codé sur deux octets et c'est le système de la pile qui est utilisé pour le stockage de ceux-ci (dernier entré, premier sorti).
Si aucun paramètre n'est entré, IX = &BFFF. IX fonctionnant comme pointeur de pile, dans le cas d'une commande 1RDSCT, 3, 5, la valeur de IX sera &BFFC et celle de A sera 2.
On obtient alors :

Adresse	Valeur
&BFFF	3
&BFFE	0
& BFFD	5
& BFFC	0

Pour l'utilisation de routines ROM demandant des paramètres dans les registres HL, DE, ou BC, vous pouvez utiliser des instructions du Z80 telles que LD E, (IX + 0), LD D, (IX + 1)......
Voici maintenant deux exemples de commandes RSX que vous pourrez utiliser si vous possédez un lecteur de disquettes. Vous pourrez les étudier, les modifier, tester leur fonctionnement pour mieux comprendre le processus.
La première de ces deux instructions est 1RDSECT, np, ns qui vous permettra, si vous possédez un lecteur de disquettes, de lire un secteur particulier d'une disquette. np doit être compris entre 0 et 39. le paramètre ns pose lui un problème car il dépend du format utilisé sur la disquette. Si celle-ci est au format DATA, &c0<ns<&CA, si c'est le format SYSTEM ou VENDOR, &40<ns&4a et si c'est le format IBM, &00<ns<&09. la routine 1WRSECT, ns, np fonctionne de la même façon que 1RDSECT sauf qu'elle écrit un secteur sur une disquette. Soyez très prudent avec cette commande qui pourrait, si vous ne faites pas attention, rendre votre disquette illisible. Travaillez seulement sur des copies.
Lorsque vous serez à l'aise dans la programmation des RSX, vous pourrez les utiliser dans un programme en Basic et vous pourrez alors diminuer la taille de celui-ci et accroître sa rapidité.

RUNTIME

Un RUNTIME est un logiciel permettant de faire tourner un programme sans avoir nécessairement besoin d'un interpréteur. Un exemple, si vous développez un logiciel sous DBASE II, que vous désirez commercialiser, il est assez difficile d'acheter un interpréteur DBASE pour chaque programme vendu, ou de demander aux utilisateurs d'acheter eux-même ce programme. Il suffit alors de demander à la société qui importe DBASE de vous vendre un nombre de RUNTIME qui feront fonctionner votre logiciel sans que le source soit accessible à l'utilisateur, et sans procédures d'installation complexes. Il existe des RUNTIME pour GEM, pour DBASE, pour le basic MALLARD (non commercialisées en FRANCE). La meilleure solution reste néanmoins (si cela existe) le compilateur, qui transforme votre programme en code machine directement exécutable. Le RUNTIME reste néanmoins une bonne solution pour des outils de développement sur lesquels il n'existe pas de compilateur ou pour, dans le cas de GEM, permettre aux utilisateurs n'ayant pas acheté l'environnement graphique de DRI d'avoir néanmoins accès au programme sans investissement supplémentaire (cas du traitement de texte EVOLUTION de la société PRIAM).



SPRITE

Dans la majorité des cas, élaborer et réaliser un programme de jeu revient à réaliser l'animation d'un objet à l'écran. Or, l'animation de ce motif entraîne souvent de grandes dificultés et un travail énorme de la part du programmeur. Un "sprite" peut être vu comme un caractère redéfini de grande dimension (ou l'agglomérat de plusieurs caractères redéfinis).

Dans vos jeux, vous pouvez vous servir de caractères simples qu'il est aisé de déplacer : de même, il est assez facile de créer des motifs utilisant l'association de quatre caractères ou plus redéfinis. Là où tout se gâte, c'est que d'une part, le motif créé

"artisanalement" est unicolore, et que d'autre part, son déplacement uniforme et harmonieux est difficile à mettre en œuvre (scrolling parfait).
C'est pourquoi de nombreux ordinateurs possèdent, comme le Commodore C.64 un synthétiseur d'images qui permet la création simplifiée d'objets multicolores et leur gestion à partir du basic (tests sur des registres de collision, de directions ...). Malheureusement l'Amstrad ne possède pas cette possibilité et les programmeurs doivent rivaliser d'ingéniosité pour animer harmonieusement des formes complexes, par l'emploi du langage machine notamment.
Les éditeurs d'utilitaires, conscients des problèmes que peut rencontrer un programmeur "moyen" quant à la création de mobiles multicolores (sprites, également appelés "lutins" ou "sylphes") ont élaboré des programmes permettant de contourner ces difficultés. Grâce à ces utilitaires, la création de sprites multicolores devient un jeu d'enfant (ou presque ...), ces derniers pouvant être stockés et réutilisés dans vos propres programmes et animés par des routines (également réincorporables) de déplacements (RSXs ...).

Il existe divers utilitaires de création de sprites. Am-Mag, dans son numéro 39, avait proposé un listing de création et de gestion simplifiée des sprites : une bonne occasion de s'y mettre !

SOURIS

Que ce petit animal mécanique est sympathique : il permet en se déplaçant sur une surface plane, de déplacer selon un mouvement proportionnel à celui de votre main, un objet sur l'écran. Lorsque vous êtes sur une zone qui correspond à vos besoins, vous cliquez pour sélectionner un objet ou une position. C'est mille fois plus pratique qu'un clavier, et parfois beaucoup plus rapide. Les souris ont conquis un large public avec les logiciels de dessin, maintenant avec des tableurs ou des traitement de texte. Mais c'est dans les domaines du dessin et de la P.A.O. que la souris fait des merveilles.
Les PC 1512 et 1640 sont équipés en standard de souris ; il en existe plusieurs pour le CPC. Celle du PCW est vendue avec l'indispensable interface.

SYSTÈME D'EXPLOITATION

Un système d'exploitation doit être écrit pour permettre à des logiciels de fonctionner sur plusieurs machines sans modification majeure (l'idéal étant sans aucune modification).

Il existe plusieurs types de système d'exploi-tation, les mono-utilisateurs (CP/M, MS-DOS, CP/M86), les mono-utilisateurs multi-tâches (DOS Plus, Concurent CPM), et les multi-utilisateurs (PROLOGUE, UNIX, XENIX).
Un système d'exploitation est constitué de plusieurs éléments invariables : un Bios, une interface utilisateur, des outils de développement. Le BIOS est la partie du système d'exploitation qui peut varier sur les machines sur lequel il est développé. En effet, un BIOS est composé de routines dont les adresses d'entrées sont invariables (dans un but de compatibilité), mais dont le contenu diffère selon l'architecture de la machine.
Prenons un exemple simple décrivant le PCW/8256 et le CPC 6128. Ces deux machines ont en commun le système d'exploitation CP/M3.0 et les logiciels sur disquette au format 3" peuvent être lus indifféremment et sans aucune modification sur l'une ou l'autre des machines. Néanmoins elles ont pour seul point commun le processeur Z80 cadencé à la même vitesse, tous les autres composants des deux cartes mères étant radicalement différents. Si nous prenons l'exemple de l'affichage vidéo, les deux gate-array sont architecturés différem-ment. Nous allons donc avoir un bios qui pour les points d'entrées vidéo sera adressé au même emplacement sur les deux machines, et dont les conditions d'entrées seront identiques (chargement de registres, registres en sortie, etc.). Néanmoins, les deux routines utiliseront ces registres différemment en fonction des processeurs qu'elles adresseront, en conservant

toutefois le même résultat à l'écran. Voici les fondements même du système d'exploitation décrits.
Une autre composante du SED, généralement incluse dans le BIOS s'intitule le DOS (pour Disque Operating System, ou système d'exploitation des disques). Cette notion, facultative avec les premiers systèmes est devenue une constante de nos jours. En effet, si les ordinateurs sur lesquels furent implantés les premiers systèmes d'exploitation avaient généralement pour mémoire de masse des lecteurs de bandes magnétiques ou des lecteurs de cartes perforées, de nos jours dont micro détient au moins une unité de disquette, voire deux ou même une unité de disque dur. Le DOS d'un système d'exploitation est donc le pack de vecteurs ou de fonctions qui permettra de gérer avec des fonctions de base telles que le formatage, la lecture de secteurs, l'écriture ou la vérification de secteurs le support physique (la diquette), et avec des fonctions plus évoluées comme l'ouverture, la lecture, l'écriture, la fermeture de fichiers, le support logique (les fichiers).
Evolution permanente de l'informatique oblige, on voit apparaître aujourd'hui une nouvelle composante du système d'exploitation que nous nommerons CDOS (Compact Disc Operating System) puisqu'elle n'a pas encore de nom, et qui permet de gérer un ou plusieurs lecteurs de CD ROM (les disques compacts). Une première implantation de CDOS devrait être effectuée sur MS-DOS par la société MICROSOFT.

L'interface utilisateur

Nous en arrivons à la seconde partie importante du système d'exploitation, l'interface utilisateur. Cette interface va permettre à l'utilisateur de saisir des commandes, de manipuler ses fichiers ou de configurer son système. Sur certains systèmes comme le CP/M, l'interface utilisateur est confondue avec le BIOS, et ne peut donc pas être modifiée, le symbole A> est la représentation de l'interface utilisateur sous CP/M), alors qu'avec des systèmes comme MS-DOS ou UNIX, il est possible de réécrire ce qui est couramment nommé un interpréteur de commande (commande SHELL du fichier CONFIG. SYS ou MS-DOS).
On assiste depuis quelque temps d'ailleurs à une généralisation d'interpréteurs de commandes plus conviviaux, et plus proches des utilisateurs, utilisant l'environnement à icônes et souris, introduit par le MACINTOSH de la société APPLE. Les principaux interpréteurs de commandes utilisant un environnement graphique actuellement sont GEM DESKTOP sous GEM de la société DIGITAL RESEARCH (utilisé sur le PC/1512 d'AMSTRAD), et WINDOWS de la société MICROSOFT. L'interpréteur de commandes est une composante très importante du système d'exploitation puisqu'il va permettre d'organiser les disquettes au mieux, de claser des fichiers et parfois, pourra être personnalisé (COMMAND.COM de MS-DOS par exemple). Dans certains cas, c'est également lui qui gérera la protection du système par mots de passe.
La dernière partie du système d'exploitation concerne les outils de développement.

Chaque système comporte sa propre gamme d'outils de développement, et plus ceux-ci sont accessibles et répandus, plus le système se vend, d'où la grande attention portée sur ce domaine par les sociétés de développement. Généralement, le système d'exploitation est livré avec plusieurs outils de développement : au moins un éditeur de texte, et un debugger. Sous CP/M 80, l'outil de développement et de debuggage s'intitule DDT. COM, et l'éditeur de texte ED ; avec CPM 3.0, l'éditeur reste ED, mais le debuggage s'intitule SID.COM (le même debugger que DDT, avec quelques commandes supplémentaires). Sous MS-DOS, les outils sont très similaires, mais comportent des noms différents : EDLIN pour l'éditeur de texte, et DEBUG pour le debugger. Notons qu'AMSTRAD, sur tous ses systèmes, remplace les infâmes et inutilisables (car remplis de caractères de contrôle) ED et EDLIN par un éditeur pleine page beaucoup plus simple d'emploi : RPED.

Les debuggers

Les debuggers, qu'ils soient DDT, SID ou DEBUG, ont plusieurs fonctions de bases qui restent constantes : un utilitaire de trace qui permet de faire fonctionner le programme examiné pas à pas avec visualisation des registres, un dump pour observer le contenu ASCII, décimal ou hexadécimal de la mémoire, un désassembleur, et un mini assembleur. Généralement, les éditeurs des systèmes d'exploitation s'empressent d'éditer en plus un assembleur (MASM pour MS-DOS, RMAC pour CP/M), des langages de développement, et des compilateurs.
Voici en quelques lignes les fondements d'un système d'exploitation, ensuite, le succès vient bien souvent grâce à une politique commerciale agressive, et un service de qualité, ce qui fit le succès de DIGITAL, et qui fait celui de MICROSOFT.

Eric CHARTON

VIDEO GATE ARRAY

Le Vidéo Gate Array (VGA) est un circuit fabriqué pour AMSTRAD et qui ne peut être vendu dans le commerce. Ce circuit gère l'écran en étant assisté par le CRTC 6845. Ce n'est pas sa seule tâche ; il est aussi chargé de réaliser la commutation des mémoires ROM / RAM, ainsi que la gestion des banques mémoires du CPC 6128. Ce circuit peut être très intéressant pour ceux qui programment en assembleur. Nous allons donc, ici, essayer de le connaître un peu mieux.

Tout d'abord, soyez très prudents si vous désirez apporter des modifications dans votre CPC. Si vous endommagez ce circuit, vous ne pourrez le changer qu'en passant par AMSTRAD, et la garantie ne jouera plus, puisque votre cher ordinateur aura été ouvert. Ceci dit, passons à l'étude logicielle qui, elle, ne souffrira pas de vos éventuelles maladresses.

Principales fonctions du processeur vidéo

Les fonctions principales de la Vidéo Gate Array sont les suivantes :

- commande des accès à la RAM ;
- commutation des mémoires ROM/RAM ;
- production des signaux vidéo et des informations RVB (Rouge, Vert, Bleu) pour le moniteur ;

- commande du mode d'écran ;
- mémorisation des couleurs d'encre.

Pour les électroniciens, sachez que ce circuit fournit aussi toutes les fréquences nécessaires, ainsi que les signaux d'exploitation des RAM dynamiques et production de l'impulsion d'interruption.

Programmation du circuit

La VGA est accessible par programme au port &7FXX (XX signifiant n'importe quelle valeur hexadécimale). C'est utile lorsque l'on désire accéder directement à ce circuit sans avoir à passer par les routines ROM. Comme vous pouvez l'imaginer, chaque bit de l'octet envoyé dans ce port à une fonction bien particulière. C'est ce que nous allons voir maintenant :
- B7 B6 (respectivement bit 7 et bit 6),
- 0 0 chargement du registre de palette.
- 0 1 chargement mémoire de palette.
- 1 0 commutation ROM et contrôle vidéo.
- 1 1 utilisé sur le CPC 6128 pour la commutation des mémoires.

Commutation ROM et contrôle vidéo

B7 = 1, B6 = 0, B5 = 0
B4 = 1 : remise à zéro du diviseur générant les interruptions.
B3 = 0 : ROM supérieure connectée.
B3 = 1 : ROM supérieure connectée.
B2 = 0 ROM inférieure connectée.
B2 = 1 : ROM inférieure connectée.
B1 : contrôle vidéo MC1.
B0 : contrôle vidéo MC0.

MC1	MC0	
0	0	Mode 0 (20 x 24).
0	1	Mode 1 (40 x 24).
1	0	Mode 2 (80 x 24).
1	1	Inutilisable.

Sur le CPC 6128, avec les valeurs suivantes, il est possible d'échanger la zone mémoire d'adresse &4000 à &7FFF (la zone, pas son contenu) de la banque 0 (banque d'origine, c'est-à-dire celle où l'on accède d'ordinaire) contre un bloc de 16 Ko de la banque 1 :
&CO : banque 0, bloc 1 (situation normale).
&C4 à &C7 : banque 1, blocs 0 à 3.
Chaque banque a une capacité mémoire de 64 Ko, et est divisée en quatre blocs de 16 Ko. Les valeurs &C1 à &C3 jouent un rôle en CP/M 3.0.

Registre et mémoire palette

Nous vous rappelons au passage qu'un registre est une mémoire de lecture - écriture.
B7 = 0, B6 = 0, B5 = 0
B4 = 0 : changement du numéro de couleur d'encre, codé par B3 à B0.
B4 = 1 : chargement du numéro de couleur de bord d'écran (B3 à B0 ne sont pas pris en compte).
B3 à B0 : donnent le numéro de l'encre (15 combinaisons possibles).
Mémoire de palette :
B7 = 0, B6 = 1, B5 = 0
B4 à B0 : 31 valeurs possibles pour le décodage de la couleur du registre de palette.
Le nombre de couleurs possibles est fonction du mode choisi.
Voici un circuit relativement simple à programmer qui peut rendre de grands services à tous ceux qui travaillent en assembleur. Soyez prudents si vous utilisez un CPC 464, ne le laissez pas allumer inutilement, pendant des heures car le circuit VGA des 464 a un gros défaut, il chauffe énormément (ce défaut a bien sûr été réduit sur les autres modèles). Laissez-le se "refroidir" de temps en temps, cela augmentera sa durée de vie et donc celle de votre CPC.



Z 80

Le Z80 a été développé dans les années 74/75 par la société ZILOG qui a ainsi conçu un processeur compatible avec le 8080 fabriqué par INTEL au début des années 70. Celui-ci à l'époque, s'était taillé une bonne part du marché faute de concurrents dans ce domaine. Les défauts de ce dernier furent éliminés et le jeu d'instructions largement étendu : ainsi naquit le Z80. Ce vieux micro-processeur a encore du succès chez les constructeurs, la preuve, il y en a un dans chaque CPC ou PCW.

Quelques données techniques

- Micro-processeur 8 bits de technologie N-MOS.
- Bus d'adresse de 16 bits (peut donc adresser 64 Ko max.).
- Signaux compatibles TTL.
- Fréquence d'horloge de 2,5, 4, 6 ou 8 MHz.
- Compatibilité logicielle avec le 8080.
- Vingt-deux registres internes dont deux registres d'index.
- Refresh automatique des RAMs dynamiques.

Les registres

Un registre est simplement une mémoire de lecture/écriture se trouvant à l'intérieur du micro-processeur. Registres du Z80 : A, F, B, C, D, E, H, L et leurs doubles A', F', B', C', D', E', H', L' ainsi que IX et IY (registres d'index), SP : stack pointeur = pointeur de pile, PC = program counter = programme compteur, I et R respectivement registre d'interruptions et registre de rafraîchissement. Les registres IX, IY, SP, PC sont des registres 16 bits (2 octets), tous les autres sont des 8 bits. Les registres A, F, B, C, D, E, H, L apparaissent en double exemplaires. Le programmeur peut choisir entre les deux jeux de registres qui ont exactement les mêmes caractéristiques. Le registre A est généralement appelé accumulateur. C'est ici que l'on obtient le résultat de toutes les opérations arithmétiques et logiques sur 8 bits. Le registre F est le registre Flag (= registre drapeau) chacun de ses bits est utilisé par le micro-processeur lors de toute opération. Un des bits de ce registre conserve une éventuelle retenue (carry en anglais) résultant par exemple d'une addition, les autres indiquent un résultat nul, un dépassement de capacité, si un résultat est négatif ... Les registres B à L ne peuvent pas uniquement être appelés séparément. On peut les regrouper pour former les registres BC, DE, HL (on peut en faire de même avec A et F pour former le registre double AF). On forme ainsi des registres 16 bits qui conviennent parfaitement à l'adressage de tableaux ainsi qu'au transfert de données en mémoire. Le registre double HL fait alors office d'accumulateur 16 bits. Le contenu du PC est placé sur le bus d'adresse comme adresse pour les mémoires externes. Lorsqu'un saut doit se produire à l'intérieur d'un programme, la nouvelle adresse est chargée dans le PC et l'exécution du programme continue à cette nouvelle adresse. Lors de l'appel d'un sous-programme, l'adresse de retour est automatiquement placée sur la pile puis rechargée dans le PC lorsque le sous-programme a été exécuté. Tant que le programmeur n'agit pas sur SP, la pile fonctionne en LIFO (Last In First Out = dernier rentré premier sorti) : on accède toujours à la pile par en haut, pour lire l'avant-dernière donnée, il faut d'abord lire la dernière. Le registre I est utilisé en liaison avec le mode d'interruption IM3. Dans ce mode, l'élément qui produit l'interruption fournit à la demande du processeur, une valeur 8 bits qui, associée à celle du registre I comme high byte (octet fort) constituent l'adresse de la routine d'interruption. Le registre R est utilisé avec le refresh exécuté automatiquement par le Z80. Chaque fois qu'une instruction a été retirée, les sept bits inférieurs du registre R sont incrémentés. Le huitième bit reste à 0 ou à 1 suivant sa programmation. Ces deux derniers registres I et R ne sont pas utilisés sur le CPC.